The Dissertation Committee for Wenqi Zhao
certifies that this is the approved version of the following dissertation:

Fast and Accurate Macromolecular Solvation Energy and Force Computations

Committee:

Chandrajit L. Bajaj, Supervisor

Leszek F. Demkowicz

Irene M. Gamba

Omar Ghattas

Thomas J.R. Hughes

Peter J. Rossky

Fast and Accurate Macromolecular Solvation Energy and Force Computations

by

Wenqi Zhao, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2008

Dedicated to my husband Liangfeng and to my parents.

Acknowledgments

I thank my supervisor, Prof. Chandrajit Bajaj, for his enormous guidance and support throughout my Ph.D. graduate research. This work would have been impossible without his insightful directions and constant encouragement.

I appreciate my Ph.D. supervising committee members, Prof. Leszek F. Demkowicz, Prof. Irene M. Gamba, Prof. Omar Ghattas, Prof. Thomas J.R. Hughes, and Prof. Peter J. Rossky for their guidance and constructive comments on my thesis.

I am grateful to Prof. Guoliang Xu who collaborated with us on the geometric modeling of the biomolecules and shared with me generous advices. I would like to thank Prof. Ron Elber and Prof. David Case for their helpful suggestions and comments on my work. I am also thankful to Prof. Arthur J. Olson and Prof. Michael Feig for providing us with the invaluable protein data sets.

I learned various knowledge in computational biology, geometry, visualization, and programming from the discussion with my colleagues in the CVC (Computational Visualization Center): Zeyun Yu, Vinay Siddavanahalli, Samrat Goswami, Jianguang Sun, Yongjie Zhang, Rezaul Chowdhury, Qin Zhang, Jose Rivera, Albert Chen, Maysam Moussalem, Andrew Gillette, Radhakr-

ishna Bettadapura, Ajay Gopinath, and others.

Many thanks go to Shan Yang for teaching me how to use Maple and sharing with me her knowledge on optimization, Andrea Hawkins, Omar al Hinai, and Henry Chang for helping me with the proof-reading of this thesis. I am grateful to the other CAM students. Thank them for their friendship, enthusiasm, and numerous suggestions, especially to Paul Bauman, David Fuentes, Nikolay Shestopalov, Shweta Bansal, Wenhao Wang, Jun Li, Kent Van Vels, Michael Harmon, and Ju Liu. I appreciate all the help that the ICES staff has provided to me, especially to Suzanne Bailey and Stephanie Rodriguez.

I owe too much to my parents for their years of efforts of bringing me up and to my sister for giving me all her love and support. Words are not enough to express my gratitude towards them. I thank my parents-in-law for their encouragement whenever I was depressed. Finally and especially, I want to thank my husband Liangfeng for his love, trust, and encouragement.

Fast and Accurate Macromolecular Solvation Energy and Force Computations

Publication No. _____

Wenqi Zhao, Ph.D.
The University of Texas at Austin, 2008

Supervisor: Chandrajit L. Bajaj

This thesis reports a comprehensive study of the electrostatic solvation energy computation for macromolecules. In the molecular dynamics (MD) simulations it is important to be able to compute the free energy of the system accurately and efficiently. The solvation energy which is dominated by the electrostatics plays a significant role in the dynamics of macromolecules in solution. The standard way of computing the electrostatic solvation energy is to solve the Poisson-Boltzmann (PB) equations. However, due to the large size of the system, the computation cost of solving the PB equation becomes a bottleneck even for the continuum implicit solvent. The alternative method is the newly developed generalized Born (GB) method which gives a good approximation to the PB calculation if the Born radii are properly computed. The computation of the Born radii is the core computation in the GB method and is laborious. In this thesis we present a novel error-bounded fast surface GB approach which significantly improves the traditional surface

GB approaches. An analytic algebraic spline model is built for the geometric model of the molecular surfaces which allows one to do the accurate computation on a coarse mesh. Based on the surface GB theory, we develop an algorithm that computes the Born radii by using the fast summation algorithm at a complexity nearly linear in terms of the number of atoms of the molecule and the number of elements on the mesh of the molecular surface. The algorithm is also extended to the electrostatic forces calculations. Finally we propose a hierarchical coarse grained (CG) model aiming at reducing the number of atoms in a macromolecule while still being able to reproduce the geometry as well as the electrostatic interactions of the atomic model.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In molecular dynamics (MD) simulations it is important to be able to compute the free energy of the system accurately and efficiently. The internal free energy of a biomolecular system is

$$U = E_{\mathrm{MM}} + \Delta G_{\mathrm{sol}}. \tag{1.1}$$

The molecular mechanical energy E_{MM} is the energy of a molecule in vacuum [3]:

$$\begin{aligned} E_{\mathrm{MM}} = &\sum_{b} k_b (r - r_{eq})^2 + \sum_{\vartheta} k_\vartheta (\vartheta - \vartheta_{eq})^2 \\ &+ \sum_{\varphi} k_\varphi (1 - \cos[n(\varphi - \varphi_{eq})]) + \sum_{i} \sum_{j<i} \left[\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{\varepsilon r_{ij}} \right]. \end{aligned}$$

The first three terms represent bonded interactions: covalent bonds, valence bonds, and torsions around bonds. The pairwise summation terms represent non-bonded interactions: the Lennard-Jones potential for van der Waals forces and the Coulomb potential for electrostatics. The force constants ($k_b,\ k_\vartheta,\ k_\varphi$), the minima ($r_{eq},\ \vartheta_{eq},\ \varphi_{eq}$), the Lennard-Jones parameters ($A_{ij},\ B_{ij}$), and the atomic charges (q_i) define the force field for the MD simulations [4]. ΔG_{sol} in (1.1) is the solvation energy affected by the solvent of the system.

The solvation effect reflects the dependence of the system on the surrounding environment. For the protein molecules, all of their activities, from folding into their native states to binding with the ligands, take place in the solvent medium, so it is of great importance to study the solvation energy to understand the chemical and physical properties that underlie the biomolecular processes.

Computer simulations of the molecular systems in which thousands of solvent molecules are explicitly taken into account can yield the most detailed information of the solvation [5]. However the inclusion of the solvent molecules severely increases the computational expense and therefore is not applicable to large systems, such as proteins. Implicit solvation models in which the solvent is viewed as a homogeneous dielectric continuum form an alternative to the explicit models [6]. The implicit model is computationally inexpensive and quantitatively reliable.

The solvation free energy includes the hydrophobic interactions, the solute-solvent van der Waals interactions, and the solute-solvent electrostatic interactions. In particular, the solvation energy is dominated by electrostatics due to the polarity of the proteins. Therefore, in many cases the hydrophobic and the van der Waals interactions are also described as continuum models based on the scaled particle theories [7] or theories of microscopic surface tension [8]. The electrostatic component of the solvation energy, also known as the polarization energy, can be computed by solving the Poisson-Boltzmann (PB) equations [9]. However, the computation cost of solving the PB equations by

the standard numerical methods of solving the partial differential equations such as the finite difference [10, 11], finite element [12, 13], and boundary element [14, 15] methods is still quite expensive for proteins or nucleic acids [2]. While the PB method continues to improve, the generalized Born (GB) approach, which is a semianalytical approximation to the PB method, has been developed and has received considerable attention [16]. As shown in Figure 1.1 is reported in [2], the GB approaches are somewhat less accurate but much faster than the PB approaches.

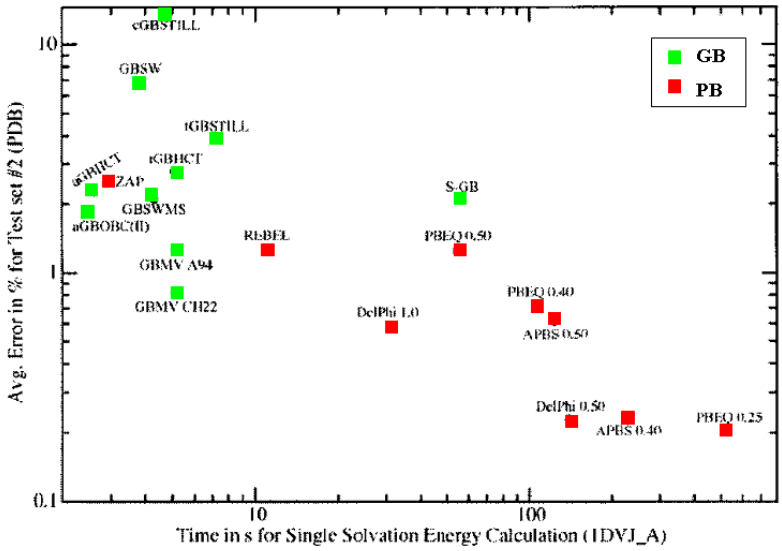


Figure 1.1: Performance of GB and PB methods measured by the average percentage error for a set of test proteins versus the time required for a calculation of the electrostatic solvation energy of a single protein 1DVJ_A [2]. Time and error axes are shown in logarithm. The labels indicate the methods used.

The GB method has shown to be a promising approach to compute the polarization energy of large systems. The size of a typical large biomolecular

complex, for example a virus, ranges from 10 nanometers to 100 nanometers. In this range, the structures are usually not a single protein but the union of tens or even thousands of individual proteins. There is much ongoing work to improve the GB method. The common goal is to reduce the computation cost as much as possible while keeping a certain accuracy. In this thesis we present a novel error-bounded fast surface GB approach which significantly improves upon the traditional surface GB approaches.

1.2 Outline of the dissertation

In Chapter 2 we present an Algebraic Spline Molecular Surface (ASMS) model. Since our GB computation is essentially based on the molecular surface, it is necessary to provide a good surface model. We introduce the steps of generating the ASMS model, discuss the properties of this surface model such as the smoothness and the error to the standard molecular surface, and apply it to the energy computation of some real proteins to demonstrate its advantages.

A comprehensive fast summation based surface GB method is introduced in Chapter 3. We first derive the main functions based on the GB theory. Then we explain how we apply the fast summation algorithm to the Born radii calculation and provide an error analysis of this approach. We test this method by implementing the energy calculation for a large set of proteins and compare its performance with other GB and PB methods.

In Chapter 4 we apply the same fast summation algorithm to the electrostatic solvation forces computation. Again we first derive the functions of

force calculation. In order to compute the derivatives of the Born radii, we use a volumetric density function, then we speed up the computation by using the fast summation method. The result of the force computation is compared with that of another GB package.

To further reduce the complexity of the energy and force computation, we build a coarse-grained (CG) model for the proteins in Chapter 5. Previous work on the CG modeling is reviewed first. Then we present a way of generating a CG model based on a hierarchical clustering of the atoms and on optimizing the geometry and GB energy of the new structure.

Chapter 6 concludes this dissertation with a summary and recommendations for future work.

Chapter 2

An Algebraic Spline Model of Molecular Surfaces

2.1 Introduction

The functions of the proteins are well determined by their three dimensional shapes. The surfaces of the molecules play a key role in their biological functions. Later in the discussion of the solvation energy computation, we will see that the solvation energy largely depends on the surface of the solute. Therefore, it is of great importance to build a good molecular surface model.

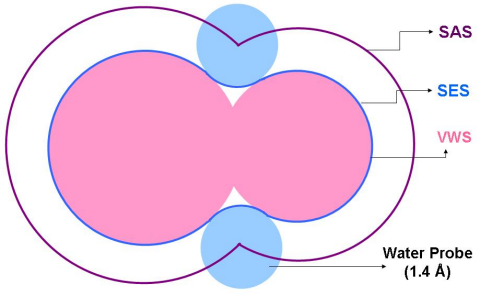


Figure 2.1: Three molecular surfaces are shown for two atoms in two dimension. The boundary of the union of balls (pink) with the van der Waals radii is the VWS. The SAS (purple) is the union of augmented van der Waals spheres with each radius enlarged by the radius of a solvent probe (light blue). The SES (the blue curve) is the boundary of all possible solvent probes that do not intersect with the interior of the VWS.

The three most well-known molecular surfaces are the van der Waals surface (VWS), the solvent accessible surface (SAS), and the solvent exclusive surface (SES) as shown in Figure 2.1 in 2D. The VWS is the union of a set of spheres with atomic van der Waals radii. The SAS is the union of augmented van der Waals spheres with each radius enlarged by the solvent probe radius (normally taken as 1.4 $\mathring{A}$) [17]. The SES (also called molecular surface or Connolly surface) is the boundary of the union of all possible solvent probes that do not intersect with the interior of the VWS [18, 19]. As described in [18], the SES consists of the convex spherical patches which are also parts of the VWS, the toroidal patches and the concave spherical patches, which are generated by the probes rolling along the intersections of neighboring atoms. The VWS causes an overestimation of the electrostatic solvation energy, while the SAS leads to an underestimation [20]. The SES is more accurate, therefore is most often used in the energy calculation. However the SES still has one significant drawback: it contains cusps when the rolling probe self-intersects, which may cause singularities in the Born radii and the force calculations.

In the energy computation, knowing the patch complexes of the molecular surface is not enough. For convenience, an analytical representation of the molecular surface is needed and the singularity should also be avoided. One way to generate such a model is to approximate the SES by an iso-contour of an analytical volumetric density function, for example, the summation of Gaussian functions [21], Fermi-Dirac switching function [22], or piecewise polynomials [20]. Techniques for fast extraction of an iso-contour of a smooth

kernel function are provided in [23, 24]. However the error of the generated isosurface could be large and result in an inaccurate energy computation. A NURBS representation for the SES is presented in [25]. Although it provides a parametric approximation to the SES, it does not solve the singularity problem. Edelsbrunner [26] defines another paradigm of a smooth surface referred to as *skin* which is based on the Delaunay and Alpha complexes of a finite set of weighted points. The *skin* model has good geometric properties such as being free of singularity and it can be decomposed into a collection of quadratic patches. Triangulation schemes based on the *skin* model are provided in [27, 28]. However when applied to the energy computation, the *skin* which is a linear approximation to the SES has to be dense enough to gain accuracy. This causes oversampling on the surface and hence makes the computation very slow. Therefore, it still remains challenging to generate a model for the molecular surface which is accurate, smooth, and computable.

In this chapter we provide a method to model the SES as piecewise algebraic spline patches with certain continuity at the boundary of the patches. Each patch has dual implicit and parametric representations. The curved implicit surfaces can be mapped onto a planar domain and therefore the two dimensional quadrature rules over a triangle can be easily applied to the patches. Moreover, because we use higher order splines to approximate the SES, a fewer number of triangles are needed compared to the piecewise linear model. The algebraic spline patches are generated based on the prism scaffold built surrounding the original triangular mesh of the SES and are implicitly defined by

simple Benstein-Bezier spline functions. Previous work on constructing piecewise spline patches within a simplical hull over a triangular mesh includes generating quadric patches [29], cubic patches [30, 31], and nonsingular and single sheeted cubic patches [32] in a tetrahedra scaffold. In this chapter, we also show that the generated algebraic spline patches are error bounded and free of singularities under certain conditions.

The rest of the chapter is organized as follows: Section 2.2 describes the details of the algebraic spline molecular surface (ASMS) generation, Section 2.3 discusses the error of ASMS, and Section 2.4 discusses the application to the energy computation and provides some examples.

2.2 Algebraic spline model

2.2.1 Algorithm Sketch

There are four main steps in the ASMS construction algorithm: (1) construct an initial triangular mesh of the SES; (2) build a prism scaffold surrounding the triangulation; (3) define a piecewise polynomial with certain continuity; (4) extract the zero-contour of the piecewise polynomial. We will the explain each step in detail in the following and discuss how to make use of the parametrization of the ASMS in the numerical integration.

2.2.2 Initial triangulation of the molecular surface

So far a lot of work has been done on the triangulation of the SES or its approximation [28, 33–36]. The ASMS generation could be applied to

any of these triangulations. In our current research, we initially represent the molecular surface as the Gaussian surface introduced in [36] which is the zero level set of the function

$$g(\mathbf{x}) = \sum_{i=1}^{M} e^{-\beta_i(||\mathbf{x}-\mathbf{x}_i||^2 - a_i^2)} - 1. \tag{2.1}$$

where M is the number of atoms in the molecule and β_i is the decay rate of the Gaussian function. From (5.3.1), one can compute the gradient

$$\nabla g(\mathbf{x}) = -2\sum_{i=1}^{M} \beta_i e^{-\beta_i(||\mathbf{x}-\mathbf{x}_i||^2 - a_i^2)}(\mathbf{x} - \mathbf{x}_i). \tag{2.2}$$

For a point $\mathbf{x}$ on the zero level set of g, the normal at $\mathbf{x}$ is given by $\mathbf{n}(\mathbf{x}) = \frac{\nabla g(\mathbf{x})}{||\nabla g(\mathbf{x})||}$. We use the suggested value $\beta_i = 2.3$ Å^{-2} for all the atoms to ensure that the zero level set of this function is close to the solvent excluded surface (SES) [21].

The triangulation of the Gaussian surface is generated by using the dual contouring method [35, 37], where a top-down octree is recursively constructed to enforce that each cell has at most one isocontour patch. Those edges whose endpoints lie on different sides of the isocontour are tagged as sign change edges. In each cube that contains a sign change edge, we compute the intersection points (and their unit normals) of the isocontour and the edges of the cube, denoted as $\mathbf{p}_i$ and $\mathbf{n}_i$, and compute the minimizer point in this cube which minimizes the quadratic error function (QEF) [38]:

$$\text{QEF}(\mathbf{x}) = \sum_i [\mathbf{n}_i \cdot (\mathbf{x} - \mathbf{p}_i)]^2$$

Since each sign change edge is shared by either four cubes (uniform grid) or three cubes (adaptive grid), connecting the minimizer points of these neighboring cubes forms a quad or a triangle that approximates the isocontour. The quads are divided into triangles to generate the pure triangular mesh. This triangulation scheme of the Gaussian surface is implemented in the software TexMol [36, 39].

2.2.3 Implicit/parametric patch generation

Given the triangulation mesh $\mathbb{T}$, let $[\mathbf{v}_i\,\mathbf{v}_j\,\mathbf{v}_k]$ be one of the triangles where $\mathbf{v}_i$, $\mathbf{v}_j$, and $\mathbf{v}_k$ are the vertices of the triangle. Suppose the unit normals of the surface at the vertices are also known, denoted as $\mathbf{n}_l,\ l = i, j, k$. Let $\mathbf{v}_l(\lambda) = \mathbf{v}_l + \lambda\mathbf{n}_l$. First we define a prism (Figure 2.2(a)):

$$D_{ijk} := \{\mathbf{p} :\ \mathbf{p} = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda),\ \lambda \in I_{ijk}\}, \tag{2.3}$$

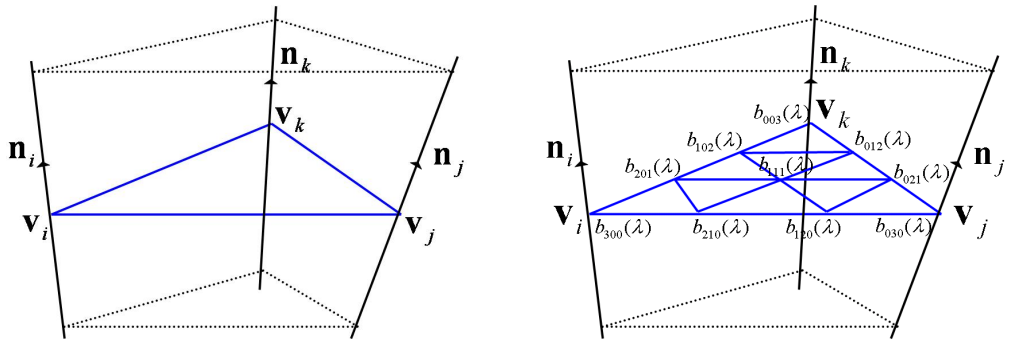
where (b_1, b_2, b_3) are the barycentric coordinates of points in $[\mathbf{v}_i\,\mathbf{v}_j\,\mathbf{v}_k]$, and I_{ijk} is the maximal open interval which satisfies: $0 \in I_{ijk}$, for any $\lambda \in I_{ijk}$, $\mathbf{v}_i(\lambda)$, $\mathbf{v}_j(\lambda)$, $\mathbf{v}_k(\lambda)$ are not collinear, and $\mathbf{n}_i$, $\mathbf{n}_j$, $\mathbf{n}_k$ point to the same side of the plane spanned by $\mathbf{v}_i(\lambda)$, $\mathbf{v}_j(\lambda)$ and $\mathbf{v}_k(\lambda)$.

Next, we define a function in the Benstein-Bezier basis over the prism D_{ijk}:

$$F(b_1, b_2, b_3, \lambda) = \sum_{i+j+k=n} b_{ijk}(\lambda) B^n_{ijk}(b_1, b_2, b_3), \tag{2.4}$$

where $B^n_{ijk}(b_1, b_2, b_3)$ is the Bezier basis

$$B^n_{ijk}(b_1, b_2, b_3) = \frac{n!}{i!j!k!} b_1^i b_2^j b_3^k.$$



(a) Prism scaffold (b) Control coefficients

Figure 2.2: (a) A prism D_{ijk} constructed based on the triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$. (b) Ten control coefficients of the cubic Bezier basis of function F.

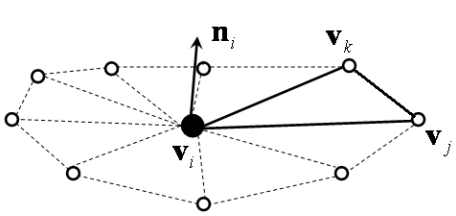
We approximate the molecular surface by the zero contour of F, denoted as S. In order to let S smooth, the degree of the Bezier basis n should be at least 3. For simplicity, in our current work we consider the case of $n = 3$. The ten control coefficients $b_{ijk}(\lambda)$ (Figure 2.2(b)) should be properly defined such that S is continuous. Next, we discuss how to set those coefficients.

Requiring S to pass through the vertices of the mesh, i.e.

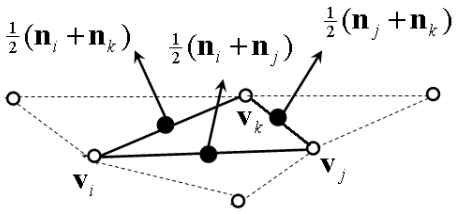
$$F(\mathbf{v}_i) = F(\mathbf{v}_j) = F(\mathbf{v}_k) = 0,$$

we get

$$b_{300}(\lambda) = b_{030}(\lambda) = b_{003}(\lambda) = \lambda. \tag{2.5}$$



(a) C^1 continuity at the vertex



(b) C^1 continuity at the midpoint of the edge

Figure 2.3: (a) S is C^1 at the vertices of the mesh and the normal of S at each vertex agrees with the given normal. (b) S is C^1 at the midpoints of the edges of the mesh.

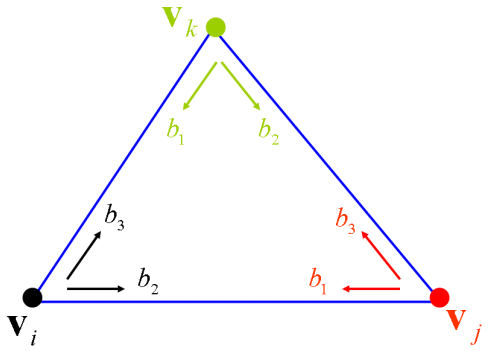


Figure 2.4: At each vertex, along the direction of the arrow, the corresponding coordinate increases.

We also require that at each vertex ∇F agrees with the given surface normal (Figure 2.3(a)), i.e.

$$\nabla F(\mathbf{v}_i) = \mathbf{n}_i, \qquad \nabla F(\mathbf{v}_j) = \mathbf{n}_j, \qquad \nabla F(\mathbf{v}_k) = \mathbf{n}_k.$$

Then at each vertex the derivatives of F along the edge directions (Figure 2.4) become

- at $\mathbf{v}_i$:

$$\frac{\partial F}{\partial b_2} = \mathbf{n}_i \cdot (\mathbf{v}_j - \mathbf{v}_i), \qquad \frac{\partial F}{\partial b_3} = \mathbf{n}_i \cdot (\mathbf{v}_3 - \mathbf{v}_1), \tag{2.6}$$

- at $\mathbf{v}_j$:

$$\frac{\partial F}{\partial b_3} = \mathbf{n}_j \cdot (\mathbf{v}_k - \mathbf{v}_j), \qquad \frac{\partial F}{\partial b_1} = \mathbf{n}_j \cdot (\mathbf{v}_1 - \mathbf{v}_2), \tag{2.7}$$

- at $\mathbf{v}_k$:

$$\frac{\partial F}{\partial b_1} = \mathbf{n}_k \cdot (\mathbf{v}_i - \mathbf{v}_k), \qquad \frac{\partial F}{\partial b_2} = \mathbf{n}_k \cdot (\mathbf{v}_2 - \mathbf{v}_3). \tag{2.8}$$

By the definition (2.4) we have

$$\begin{aligned} \frac{\partial F}{\partial b_2} =& -3b_{300}(\lambda)b_1^2 + 3b_{210}(\lambda)(b_1^2 - 2b_1b_2) + 3b_{120}(\lambda)(2b_1b_2 - b_2^2) \\ &+ 3b_{030}(\lambda)b_2^2 + 6b_{021}(\lambda)b_2b_3 + 3b_{012}(\lambda)b_3^2 \\ &- 3b_{102}(\lambda)b_3^2 - 6b_{201}(\lambda)b_1b_3 + 6b_{111}(\lambda)(b_1b_3 - b_2b_3). \end{aligned} \tag{2.9}$$

Noticing that at $\mathbf{v}_i$, $b_1 = 1$, $b_2 = b_3 = 0$, so

$$\frac{\partial F}{\partial b_2}|_{(1,0,0)} = -3b_{300}(\lambda) + 3b_{210}(\lambda). \tag{2.10}$$

From (2.6) and (2.10), we get

$$b_{210}(\lambda) = \lambda + \frac{1}{3}\mathbf{n}_i \cdot (\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda)). \tag{2.11}$$

Similarly we may derive the definition of $b_{120}(\lambda)$, $b_{201}(\lambda)$, $b_{102}(\lambda)$, $b_{021}(\lambda)$, and $b_{012}(\lambda)$. So far we have defined 9 coefficients in Figure 2.5(a), for the remaining

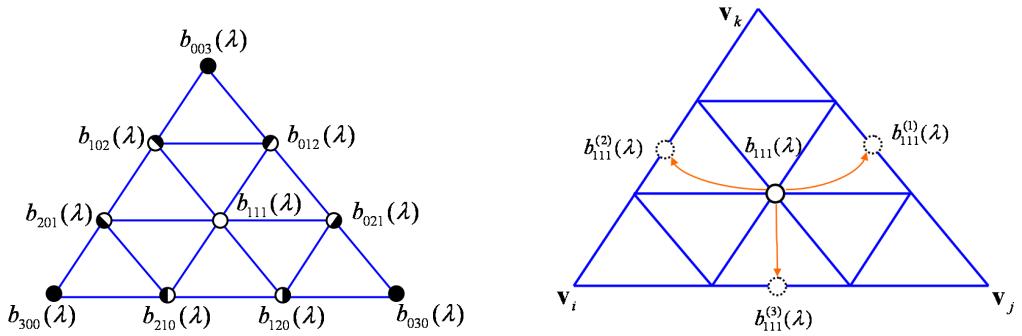


Figure 2.5: (a) The coefficients marked as solid circles are defined by the interpolation of the surface S at the vertices. The coefficients marked as half solid circles are defined by the C^1 continuity of S at the vertices. (b) The remaining degree of freedom marked as the hollow circle is defined as the weighted sum of three coefficients $b_{111}^{(1)}(\lambda)$, $b_{111}^{(2)}(\lambda)$, $b_{111}^{(3)}(\lambda)$, each of which is defined such that the S is C^1 continuous at the midpoints of the edges.

coefficient $b_{111}(\lambda)$, we define by using the side-vertex scheme [40] to increase the degrees of freedom (Figure 2.5(b)):

$$b_{111}(\lambda) = w_1 b_{111}^{(1)}(\lambda) + w_2 b_{111}^{(2)}(\lambda) + w_3 b_{111}^{(3)}(\lambda), \tag{2.12}$$

where

$$w_i = \frac{b_j^2 b_k^2}{b_2^2 b_3^2 + b_1^2 b_3^2 + b_1^2 b_2^2}, \qquad i = 1, 2, 3,\ i \neq j \neq k.$$

$b_{111}^{(1)}(\lambda)$, $b_{111}^{(2)}(\lambda)$ and $b_{111}^{(3)}(\lambda)$ will be defined such that the C^1 continuity is achieved at the midpoint of the edge $\mathbf{v}_j\mathbf{v}_k$, $\mathbf{v}_i\mathbf{v}_k$, and $\mathbf{v}_i\mathbf{v}_j$, respectively (Figure 2.3(b)). Let us consider the edge $\mathbf{v}_i\mathbf{v}_j$. Recall that any point $\mathbf{p} = (x, y, z)$ in D_{ijk} can be represented by

$$(x, y, z)^{\mathrm{T}} = b_1 \mathbf{v}_i(\lambda) + b_2 \mathbf{v}_j(\lambda) + b_3 \mathbf{v}_k(\lambda). \tag{2.13}$$

Therefore, by differentiating both sides of (2.13) with respect to x, y and z, respectively, we get

$$I_3 = \begin{pmatrix} \frac{\partial b_1}{\partial x} & \frac{\partial b_2}{\partial x} & \frac{\partial \lambda}{\partial x} \\ \frac{\partial b_1}{\partial y} & \frac{\partial b_2}{\partial y} & \frac{\partial \lambda}{\partial y} \\ \frac{\partial b_1}{\partial z} & \frac{\partial b_2}{\partial z} & \frac{\partial \lambda}{\partial z} \end{pmatrix} \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^{\mathrm{T}} \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^{\mathrm{T}} \\ (b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k)^{\mathrm{T}} \end{pmatrix}, \tag{2.14}$$

where I_3 is a 3×3 unit matrix. Denote

$$T := \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^{\mathrm{T}} \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^{\mathrm{T}} \\ (b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k)^{\mathrm{T}} \end{pmatrix}, \tag{2.15}$$

and let $A = \mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda)$, $B = \mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda)$ and $C = b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k$, then $T = (A\ B\ C)^{\mathrm{T}}$. From (2.14) we have

$$\begin{pmatrix} \frac{\partial b_1}{\partial x} & \frac{\partial b_2}{\partial x} & \frac{\partial \lambda}{\partial x} \\ \frac{\partial b_1}{\partial y} & \frac{\partial b_2}{\partial y} & \frac{\partial \lambda}{\partial y} \\ \frac{\partial b_1}{\partial z} & \frac{\partial b_2}{\partial z} & \frac{\partial \lambda}{\partial z} \end{pmatrix} = T^{-1} = \frac{1}{\det(T)} \left(B \times C,\ C \times A,\ A \times B\right). \tag{2.16}$$

According to (2.4), at the midpoint of $\mathbf{v}_i\mathbf{v}_j$, $(b_1, b_2, b_3) = (\frac{1}{2}, \frac{1}{2}, 0)$, we have

$$\begin{pmatrix} \frac{\partial F}{\partial b_1} \\ \frac{\partial F}{\partial b_2} \\ \frac{\partial F}{\partial \lambda} \end{pmatrix} = \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^{\mathrm{T}} \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^{\mathrm{T}} \\ (\mathbf{n}_i + \mathbf{n}_j)^{\mathrm{T}}/2 \end{pmatrix} \left(\frac{\mathbf{n}_i + \mathbf{n}_j}{4}\right) + \begin{pmatrix} \frac{3}{2}(b_{210}(\lambda) - b_{111}(\lambda)) \\ \frac{3}{2}(b_{120}(\lambda) - b_{111}(\lambda)) \\ \frac{1}{2} \end{pmatrix}.$$

By (2.12), at $(b_1, b_2, b_3) = (\frac{1}{2}, \frac{1}{2}, 0)$ we have $b_{111}(\lambda) = b^{(3)}_{111}(\lambda)$. Therefore, the gradient at $(\frac{1}{2}, \frac{1}{2}, 0)$ is

$$\begin{aligned} \nabla F &= T^{-1} \left(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \frac{\partial F}{\partial \lambda}\right)^{\mathrm{T}} \\ &= \frac{\mathbf{n}_i + \mathbf{n}_j}{4} + \frac{1}{2\det(T)}[3(b_{210} - b^{(3)}_{111})B \times C + \ 3(b_{120} - b^{(3)}_{111})C \times A + A \times B] \end{aligned} \tag{2.17}$$

Define vectors

$$\begin{aligned}
&\mathbf{d}_1(\lambda) = \mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda) = B - A,\\
&\mathbf{d}_2(b_1, b_2, b_3) = b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k = C,\\
&\mathbf{d}_3(b_1, b_2, b_3, \lambda) = \mathbf{d}_1 \times \mathbf{d}_2 = B \times C + C \times A. \qquad (2.18)
\end{aligned}$$

Let

$$\mathbf{c} = C(\frac{1}{2}, \frac{1}{2}, 0), \qquad (2.19)$$

$$\mathbf{d}_3(\lambda) = \mathbf{d}_3(\frac{1}{2}, \frac{1}{2}, 0, \lambda) = B \times \mathbf{c} + \mathbf{c} \times A. \qquad (2.20)$$

Let $\nabla F = \nabla F(\frac{1}{2}, \frac{1}{2}, 0)$. In order to have C^1 continuity at $(\frac{1}{2}, \frac{1}{2}, 0)$, we should have $\nabla F \cdot \mathbf{d}_3(\lambda) = 0$. Therefore, by (2.17) and (2.20), we have

$$b_{111}^{(3)}(\lambda) = \frac{\mathbf{d}_3(\lambda)^{\mathrm{T}}(3b_{210}(\lambda)B \times \mathbf{c} + 3b_{120}(\lambda)\mathbf{c} \times A + A \times B)}{3||\mathbf{d}_3(\lambda)||^2}. \qquad (2.21)$$

Similarly, we may define $b_{111}^{(1)}(\lambda)$ and $b_{111}^{(2)}(\lambda)$.

Now the function $F(b_1, b_2, b_3, \lambda)$ is well defined. The next step is to extract the zero level set S. Given the barycentric coordinates (b_1, b_2, b_3) of a point in the triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$, we find the corresponding λ by solving the equation $F(b_1, b_2, b_3, \lambda) = 0$ for λ and this could be done by the Newton's method. Then we may get the corresponding point on S as

$$(x, y, z)^{\mathrm{T}} = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda). \qquad (2.22)$$

2.2.4 Normal and curvature

Let $\mathcal{T} = T^{-1}$. The surface normal $n = \frac{\nabla F}{||\nabla F||}$, where $\nabla F = \mathcal{T}\left(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \frac{\partial F}{\partial \lambda}\right)^{\mathrm{T}}$. The Hessian matrix of F is

$$H = \begin{pmatrix} F_{xx} & F_{xy} & F_{xz} \\ F_{xy} & F_{yy} & F_{yz} \\ F_{xz} & F_{yz} & F_{zz} \end{pmatrix} = \begin{pmatrix} \mathcal{T}_x & \mathcal{T}_y & \mathcal{T}_z \end{pmatrix}_{3\times 9} \begin{pmatrix} \nu & & \\ & \nu & \\ & & \nu \end{pmatrix}_{9\times 3} + \mathcal{T} M \mathcal{T}^{\mathrm{T}},$$

where the vector $\nu = \left(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \frac{\partial F}{\partial \lambda}\right)^{\mathrm{T}}$,

$$\mathcal{T}_x = \frac{\partial \mathcal{T}}{\partial x} = \begin{pmatrix} \frac{\partial^2 b_1}{\partial x^2} & \frac{\partial^2 b_2}{\partial x^2} & \frac{\partial^2 \lambda}{\partial x^2} \\ \frac{\partial^2 b_1}{\partial x \partial y} & \frac{\partial^2 b_2}{\partial x \partial y} & \frac{\partial^2 \lambda}{\partial x \partial y} \\ \frac{\partial^2 b_1}{\partial x \partial z} & \frac{\partial^2 b_2}{\partial x \partial z} & \frac{\partial^2 \lambda}{\partial x \partial z} \end{pmatrix},$$

$$\mathcal{T}_y = \frac{\partial \mathcal{T}}{\partial y} = \begin{pmatrix} \frac{\partial^2 b_1}{\partial x \partial y} & \frac{\partial^2 b_2}{\partial x \partial y} & \frac{\partial^2 \lambda}{\partial x \partial y} \\ \frac{\partial^2 b_1}{\partial y^2} & \frac{\partial^2 b_2}{\partial y^2} & \frac{\partial^2 \lambda}{\partial y^2} \\ \frac{\partial^2 b_1}{\partial y \partial z} & \frac{\partial^2 b_2}{\partial y \partial z} & \frac{\partial^2 \lambda}{\partial y \partial z} \end{pmatrix},$$

$$\mathcal{T}_z = \frac{\partial \mathcal{T}}{\partial z} = \begin{pmatrix} \frac{\partial^2 b_1}{\partial x \partial z} & \frac{\partial^2 b_2}{\partial x \partial z} & \frac{\partial^2 \lambda}{\partial x \partial z} \\ \frac{\partial^2 b_1}{\partial y \partial z} & \frac{\partial^2 b_2}{\partial y \partial z} & \frac{\partial^2 \lambda}{\partial y \partial z} \\ \frac{\partial^2 b_1}{\partial z^2} & \frac{\partial^2 b_2}{\partial z^2} & \frac{\partial^2 \lambda}{\partial z^2} \end{pmatrix},$$

and

$$M = \begin{pmatrix} F_{b_1 b_1} & F_{b_1 b_2} & F_{b_1 \lambda} \\ F_{b_1 b_2} & F_{b_2 b_2} & F_{b_2 \lambda} \\ F_{b_1 \lambda} & F_{b_2 \lambda} & F_{\lambda\lambda} \end{pmatrix}.$$

Knowing the Hessian matrix, one can easily compute the *mean* curvature of the ASMS:

$$\begin{aligned} \mathcal{H} &= -\frac{1}{2}\mathrm{div}\left(\frac{\nabla F}{||\nabla F||}\right) \\ &= \big(F_x^2(F_{yy}+F_{zz}) + F_y^2(F_{zz}+F_{xx}) + F_z^2(F_{xx}+F_{yy}) \\ &\quad -2F_xF_yF_{xy} - 2F_yF_zF_{yz} - 2F_zF_xF_{xz}\big)/||\nabla F||^3, \end{aligned} \tag{2.23}$$

and the *Gaussian* curvature of the ASMS:

$$\begin{aligned}\mathcal{K} =& - ||\nabla F||^{-4} \det \begin{pmatrix} \nabla^2 F & \nabla F \\ \nabla F^T & 0 \end{pmatrix} \\ =& (2F_x F_y (F_{xz} F_{yz} - F_{xy} F_{zz}) + 2F_y F_z (F_{xy} F_{xz} - F_{yz} F_{xx}) + \\ & 2F_z F_x (F_{yz} F_{xy} - F_{xz} F_{yy})) / ||\nabla F||^4. \end{aligned} \tag{2.24}$$

2.2.5 Smoothness

Theorem 2.2.1. The ASMS S is C^1 at the vertices of the mesh and the midpoints of the edges of mesh.

Theorem 2.2.2. S is C^1 everywhere if every edge $\mathbf{v}_i\mathbf{v}_j$ of the mesh satisfies $\mathbf{n}_i \cdot (\mathbf{v}_i - \mathbf{v}_j) = \mathbf{n}_j \cdot (\mathbf{v}_j - \mathbf{v}_i)$.

Theorem 2.2.3. S is C^1 everywhere if the unit normals at the vertices of the mesh are the same.

Proofs of the theorems are shown in Appendix A.

2.3 Error analysis of the ASMS model

In order to show the error of S to the true surface S_0, we do a test on some typical surfaces $S_0 := \{(x, y, z) : z = f(x, y), (x, y) \in [0, 1]^2\}$ which are considered as the true surfaces (Table 2.1). We generate a triangulation mesh over the true surface with the maximum edge length h being 0.1. Based on the mesh, we construct the ASMS model S. The error of S to S_0 is defined as

$\max \frac{||\mathbf{p}-\mathbf{q}||}{||\mathbf{q}||}$, where $\mathbf{p} \in S$, $\mathbf{q} \in S_0$, and $\mathbf{p}$ and $\mathbf{q}$ have the same (b_1, b_2, b_3) volume coordinates but different λ coordinates. We sample $(\mathbf{p},\ \mathbf{q})$ on the surfaces and compute the maximum relative error. For the point pair $\mathbf{p}(b_1, b_2, b_3, \lambda_p)$

Function $(x, y) \in [0, 1]^2$	$\max\{\frac{\|\mathbf{p}-\mathbf{q}\|}{\|\mathbf{q}\|}\}$	C
$z = 0$	0	0
$z = x^2 + y^2$	2.450030e-05	1.010636e-2
$z = x^3 + y^3$	1.063699e-04	2.610113e-2
$z = e^{-\frac{1}{4}[(x-0.5)^2+(y-0.5)^2]}$	5.286856e-07	6.288604e-5
$z = 1.25 + \frac{\cos(5.4y)}{6+6(3x-1)^2}$	2.555683e-04	4.58608e-2
$z = \tanh(9y - 9x)$	1.196519e-02	1.896754e-1
$z = \sqrt{1 - x^2 - y^2}$	8.614969e-05	1.744051e-1 (h^4)
$z = [(2 - \sqrt{1 - y^2})^2 - x^2]^{1/2}$	1.418242e-05	1.748754e-02

Table 2.1: For some typical explicit surfaces (column 1), we compute the maximum relative error (the ratio of the Euclidian distance between $\mathbf{p}$ and $\mathbf{q}$ to the norm of $\mathbf{q}$) in column 2 with $h = 0.1$, $(x, y) \in [0, 1]^2$. In column 3, we show the limit of $\frac{|\lambda-\lambda'|}{h^3}$ as $h \downarrow 0$ denoted as C which verifies the rate of convergence of S as we claimed. For certain cases, like $z = \sqrt{1 - x^2 - y^2}$, the rate of convergence can be as good as $O(h^4)$.

and $\mathbf{q}(b_1, b_2, b_3, \lambda_q)$ defined above, we prove that their Euclidean distance is bounded by the difference of their λ coordinates.

Lemma 2.3.1. The error of the approximation point $\mathbf{p}$ to the true point $\mathbf{q}$ is bounded by $|\lambda_p - \lambda_q|$.

Proof of Lemma 2.3.1:

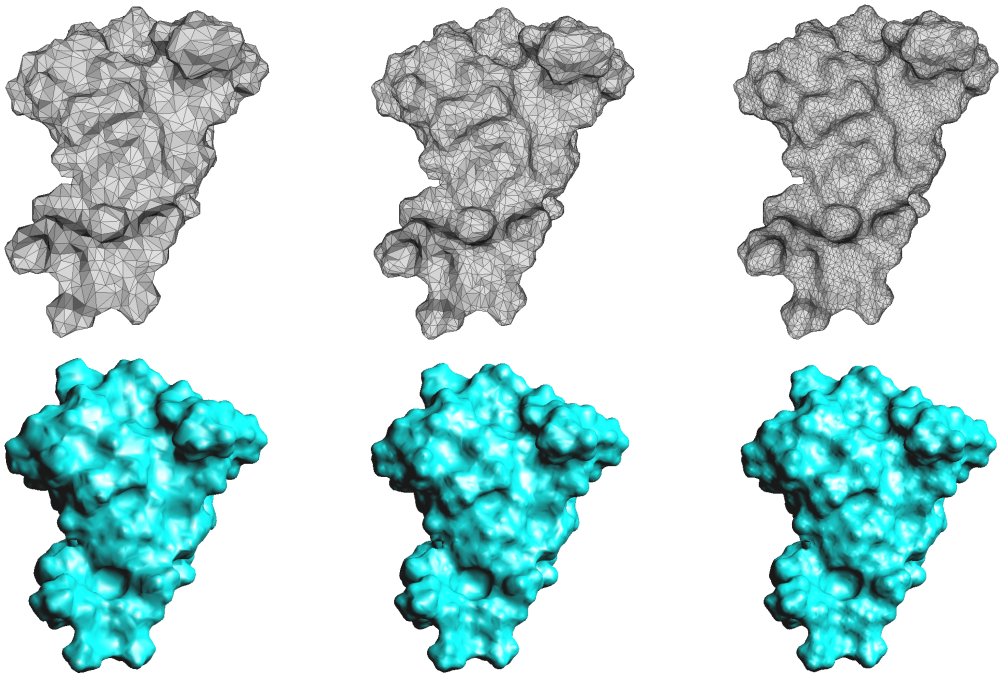
$$\begin{aligned}||\mathbf{p}-\mathbf{q}|| &\leq b_1||\mathbf{v}_i(\lambda_p)-\mathbf{v}_i(\lambda_q)|| + b_2||\mathbf{v}_j(\lambda_p)-\mathbf{v}_j(\lambda_q)|| + b_3||\mathbf{v}_k(\lambda_p)-\mathbf{v}_k(\lambda_q)|| \\ &\leq |\lambda_p-\lambda_q|(b_1||\mathbf{n}_i|| + b_2||\mathbf{n}_j|| + b_3||\mathbf{n}_k||) \\ &= |\lambda_p-\lambda_q|\end{aligned}$$

□

To study the rate of convergence of S to S_0, we gradually refine the initial mesh. Since the error is bounded by $|\lambda_p - \lambda_q|$, we compute the ratio of the maximum difference of λ_p and λ_q to h, h^2, h^3, and so forth. As h decreases, we check if the ratio converges, which allows us to know the highest rate of convergence of S to S_0. For most of the test functions in Table 2.1, we observe that S converges to S_0 as fast as $O(h^3)$. We also observe that for the case $z = \sqrt{1-x^2-y^2}$, the rate of convergence reaches $O(h^4)$. We show the limit of the ratio $\frac{|\lambda-\lambda'|}{h^3}$, denoted as C, as h goes to zero, in Table 2.1. Hence we draw the following claim:

Claim: Let h be the maximum side length of triangulation mesh $\mathbb{T}$, $\mathbf{p}$ be the point on the ASMS, $\mathbf{q}$ be the corresponding point on the true surface, then $\mathbf{p}$ converges to $\mathbf{q}$ at the rate of $O(h^3)$. i.e. There exists a constant C such that $||\mathbf{p}-\mathbf{q}|| \leq Ch^3$.

We generat the ASMS for the real proteins based on different size of meshes, as shown in Figure 2.6 for protein 1ML0 as an example. We compute the error



4600 Triangles 9216 Triangles 18434 Triangles

Figure 2.6: The top row is the triangulation of the SES of protein 1ML0 with different number of triangles. The bottom row is the ASMS generated from the above corresponding triangulation.

of the ASMS, generated from the triangular meshes of different sizes, to the SES, as shown in Table 2.2 for three proteins: 1GCQ (843 atoms), 1ML0 (1051 atoms), and 1KKL (1276 atoms). Here the SES is modelled as the Gaussian surface introduced in Section 2.2.2. The error ε_{max} is defined as the one-way Hausdorff distance from the ASMS to the SES: $\varepsilon_{max} = \max_{\mathbf{p}\in\text{ASMS}} \min_{\mathbf{q}\in\text{SES}} ||\mathbf{p}-\mathbf{q}||$. As we see in the table, the errors are small and decrease rapidly as the initial triangulation becomes more and more dense.

1GCQ		1ML0		1KKL	
No. of Δs	ε_{max}	No. of Δs	ε_{max}	No. of Δs	ε_{max}
16,312	0.266069	18,400	0.233949	19,968	0.260418
32,624	0.142149	36,864	0.142380	39,544	0.134689
65,456	0.082550	73,736	0.083895	79,096	0.085855

Table 2.2: We compute the error of the ASMS to the SES for protein 1GCQ, 1Ml0, and 1KKL. For each protein, the first column is the number of triangles of the initial mesh based on which the ASMS is generated and the second column is the error of the ASMS to the SES, which is defined as $\varepsilon_{max} = \max_{\mathbf{p}\in\text{ASMS}} \min_{\mathbf{q}\in\text{SES}} ||\mathbf{p}-\mathbf{q}||$.

2.4 Application

2.4.1 Numerical integration

The integrations over the ASMS can be easily computed by using the parametric representation of the ASMS. Suppose we are going to compute $\int_S f(\mathbf{x})\ \mathrm{d}S$, where S is modeled as the ASMS. We decompose the entire surface S into patches $\{S_j\}$ with S_j being the AMSM generated over triangle j, then we have

$$\int_S f(\mathbf{x})\ \mathrm{d}S = \sum_j \int_{S_j} f(\mathbf{x})\ \mathrm{d}S. \tag{2.25}$$

For any point $\mathbf{x} = (x, y, z) \in S_j$, by the inverse map of (2.22), one can always map $\mathbf{x}$ to a point in triangle j, denoted as σ_j, with the barycentric coordinates (b_1, b_2, b_3) where $b_3 = 1 - b_1 - b_2$. Therefore, x, y, z can be parameterized in terms of (b_1, b_2):

$$x = x(b_1, b_2,), \qquad y = y(b_1, b_2), \qquad z = z(b_1, b_2)$$

Replacing (x, y, z) with (b_1, b_1, b_3) in (2.25) and setting

$$g(b_1, b_2) = f(x(b_1, b_2), y(b_1, b_2), z(b_1, b_2)),$$

we get

$$\begin{aligned}\int_{S_j} f(\mathbf{x})\ dS &= \int_{\sigma_j} g(b_1, b_2)|\frac{\partial \mathbf{x}}{\partial b_1} \times \frac{\partial \mathbf{x}}{\partial b_2}|\ \mathrm{d}b_1\mathrm{d}b_2 \\ &= \int_{\sigma_j} g(b_1, b_2)\sqrt{EG - F^2}\ \mathrm{d}b_1\mathrm{d}b_2, \end{aligned} \tag{2.26}$$

where

$$\begin{aligned} E &= (\frac{\partial x}{\partial b_1})^2 + (\frac{\partial y}{\partial b_1})^2 + (\frac{\partial z}{\partial b_1})^2, \\ F &= \frac{\partial x}{\partial b_1}\frac{\partial x}{\partial b_2} + \frac{\partial y}{\partial b_1}\frac{\partial y}{\partial b_2} + \frac{\partial z}{\partial b_1}\frac{\partial z}{\partial b_2}, \\ G &= (\frac{\partial x}{\partial b_2})^2 + (\frac{\partial y}{\partial b_2})^2 + (\frac{\partial z}{\partial b_2})^2. \end{aligned}$$

Applying the Gaussian quadrature to (2.26), we get

$$\int_{\sigma_i} g(b_1, b_2)\sqrt{EG - F^2}\ \mathrm{d}b_1\mathrm{d}b_2 \approx \sum_{k=1}^{n} W_k g(b_1^k, b_2^k)\sqrt{EG - F^2}|_{b_1^k, b_2^k}, \tag{2.27}$$

where (b_1^k, b_2^k, b_3^k) and W_k are the Gaussian integration nodes and weights on the triangles.

One important application of the integration over the ASMS is computing the area of ASMS where f in (2.25) is replaced by one. The ASMS can also be easily applied to the boundary element method to solve PDEs.

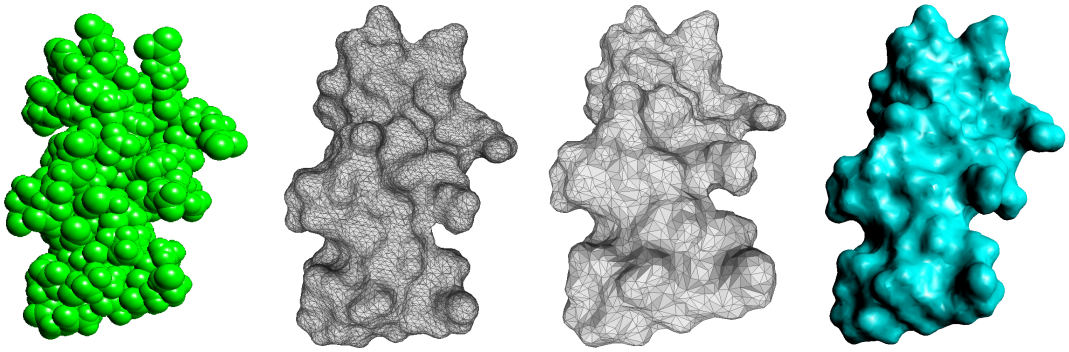
2.5 Application to the biomolecular energy computation

In this section, we apply the ASMS model to the GB electrostatic solvation energy computations of the example proteins 1HIA (693 atoms), 1CGI (852 atoms), and 1PPE (436 atoms). The ASMS models S for the proteins are generated based on the initial mesh with different numbers of triangles (Table 2.3). In Figure 2.7 we show a fine and a coarse mesh of 1HIA and the ASMS generated from the coarse mesh. Figure 2.8 shows the ASMS of 1CGI and 1PPE generated from the decimated triangulations. To demonstrate the advantage of the ASMS in the energy calculation, we compute the polarization energy G_{pol} based on both the ASMS and the piecewise linear surfaces. For all the computations, a three-point Gaussian quadrature rule over a triangle [1] is used for the numerical integration in (2.27) when computing the Born radii. We consider the running time as the total time cost of computing the integration nodes over the surfaces, computing the Born radii, and evaluating G_{pol}. If we regard the energy computed from the dense mesh as accurate, as we see from Table 2.3, the G_{pol} computed from the coarse piecewise linear model has a large error, however for the coarse ASMS model, it is very close to the dense mesh result but with less time. On the other hand, to get a energy result of the same accuracy, fewer number of triangles are needed for the ASMS model than the piecewise linear model. For example, for the protein 1CGI, the G_{pol} computed from the ASMS with 3674 triangles is -1394.227 kcal/mol. However to get a similar result, 8712 triangles are needed for the piecewise linear model.

Therefore, the ASMS model is much more efficient in the energy computation than trivial piecewise linear models.

Protein ID	No. of Triangles	G_{pol} (kcal/mol)		Timing (s)	
		PL	AS	PL	AS
1CGI	29108	-1371.741894	-1343.1496	39.64	40.31
	8712	-1399.194841	-1346.2230	12.94	12.64
	3674	-1678.444735	-1394.2270	7.40	6.11
1HIA	27480	-1361.226603	-1340.6384	30.23	31.18
	7770	-1389.017538	-1347.8067	9.43	9.93
	3510	-1571.890827	-1388.4665	5.21	5.21
1PPE	24244	-835.563905	-825.3252	17.27	18.26
	6004	-852.713039	-828.2158	5.09	5.39
	2748	-933.956234	-845.5085	2.74	3.27

Table 2.3: Comparison of the electrostatic solvation energy computed by the piecewise linear surface (PL) and the ASMS. Consider the energy computed from the dense mesh as accurate. With fewer number of triangles used, energy error of the ASMS is smaller than the PL. To get a energy of the same accuracy, fewer number of triangles are needed for the ASMS model than the PL. The running time contains the time cost of computing the integration nodes over the surfaces, computing the Born radii, and evaluating G_{pol}.



(a) (b) (c) (d)

Figure 2.7: Molecular models of a protein(1HIA). (a) is The atomic model. (b) is the initial dense mesh of the SES (27480 triangles). (c) is the decimated mesh of the SES model (7770 triangles). (d) is the ASMS (7770 patches) generated from (c).

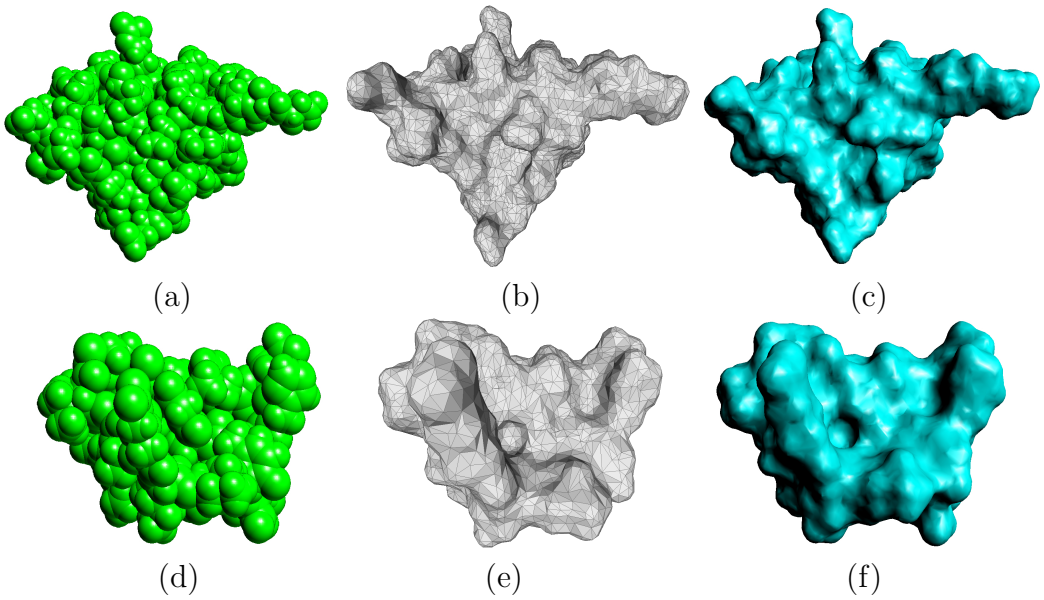


Figure 2.8: The top row are the models of 1CGI and the bottom row are the models of 1PPE. (a) and (d) are the atomic structures of the proteins. (b) and (e) are the decimated triangular meshes of the proteins with 8712 triangles and 6004 triangles, respectively. (c) and (f) are the ASMS models generated from (b) and (e), respectively.

Chapter 3

Solvation Energetics Computation

3.1 Introduction

As discussed in Chapter 1, the solvation energy consists of the energy of the hydrophobic interaction which associates with forming a cavity in the solvent, the solute-solvent van der Waals interaction, and the solute-solvent polarization energy) due to the charges of the solute:

$$\Delta G_{\text{sol}} = G_{\text{cav}} + G_{\text{vdw}} + G_{\text{pol}}. \tag{3.1.1}$$

G_{cav} and G_{vdw} are also known as the non-polar energy terms in (3.1.1). In [41–45], the non-polar energies are linearly related to the solvent-accessible surface area:

$$G_{\text{cav}} + G_{\text{vdw}} = \sum_j \gamma_j A_j \tag{3.1.2}$$

where the sum is over all solute atoms, γ_j is the atomic surface tension, A_j is the atomic solvent accessible surface area. For simplicity, γ_j is taken as the same parameter for all the atoms (e.g. 7.2 cal/(mol Å^2) in [46]). Then non-polar energy is linearly proportional to the area of the solvent accessible surface of the molecule. There are also cases, especially for some small molecules, that the non-polar energies are linearly related to the solvent excluded volume [45, 47].

The solvation energy is dominated by the polarization energy which reflects the change of the electrostatic energy due to the induced polarization in the solvent. So the electrostatic component of the solvation energy is

$$G_{\text{pol}} = \frac{1}{2} \int \phi_{\text{reaction}}(\mathbf{r}) \rho(\mathbf{r}) \, \mathrm{d}\mathbf{r} \tag{3.1.3}$$

where $\rho(\mathbf{r})$ is the charge density at position $\mathbf{r}$, the reaction field $\phi_{\text{reaction}} = \phi_{\text{solvent}} - \phi_{\text{gas-phase}}$, ϕ_{solvent} and $\phi_{\text{gas-phase}}$ denote the electrostatic potential of the system in the gas-phase and in a solvent environment, respectively.

One can compute the electrostatic potentials ϕ_{solvent} and $\phi_{\text{gas-phase}}$ by solving the PB equations. But as discussed in Chapter 1, the PB methods are often computationally too expensive for large biomolecules and are often replaced by the GB methods. Even though the GB computation is much faster than the PB model, computing the Born radii of the solute atoms is still very time-consuming which limits many of the MD simulations of the protein activities, such as the protein folding, to small time scales and small length scales [48].

In this chapter we mainly develop a method to efficiently compute the Born radii of the atoms by using the fast summation algorithm based on the nonequispaced fast Fourier transform (NFFT). An efficient way of sampling the quadrature points on the nonlinear patches is given with the aids of using the ASMS introduced in Chapter 2. We also show that the error of the Born radius calculation is controlled by the size of the triangulation mesh and the regularity of the periodic function used in the fast summation algorithm. The time

complexity of the new GB energy computation is reduced from the original $O(MN)$ to the nearly linear time complexity $O(N + M + n^3 \log n)$.

The content of this chapter is organized as follows: in Section 3.2 we give an introduction to the generalized Born theory, in Section 3.3 we review the previous work of the effective Born radii computation. We provide the fast summation based surface GB methods and its error analysis in Section 3.4 and demonstrate the test results in Section 3.5.

3.2 Generalized Born energy

The basic idea of Generalized Born method dates back to the work of Max Born in 1920 who first gave the formula of computing the electrostatic solvation free energy of a single ion. In his model, the ion is represented as a ball of radius a containing a point charge q at its center $\mathbf{c}$. Inside the ion, the *dielectric constant* (a physical quantity which describes the ability of a material to polarize in response to an electric field) is assumed to be one. In the gas phase, the electric potential of the ion is

$$\phi_{\text{gas-phase}}(\mathbf{r}) = \frac{q}{|\mathbf{r} - \mathbf{c}|}. \tag{3.2.1}$$

By immersing the ion in a solvent with the dielectric constant ε (usually water has $\varepsilon = 80$), the electric potential becomes

$$\phi_{\text{solvent}}(\mathbf{r}) = \begin{cases} \frac{q}{|\mathbf{r}-\mathbf{c}|} - (1 - \frac{1}{\varepsilon})\frac{q}{a} & |\mathbf{r} - \mathbf{c}| < a \\ \frac{q}{\varepsilon|\mathbf{r}-\mathbf{c}|} & |\mathbf{r} - \mathbf{c}| \geq a \end{cases}. \tag{3.2.2}$$

The reaction field is

$$\phi_{\text{reaction}}(\mathbf{r}) = \begin{cases} -(1-\frac{1}{\varepsilon})\frac{q}{a} & |\mathbf{r}-\mathbf{c}| < a \\ (\frac{1}{\varepsilon}-1)\frac{q}{|\mathbf{r}-\mathbf{c}|} & |\mathbf{r}-\mathbf{c}| \geq a \end{cases}, \tag{3.2.3}$$

and the electrostatic solvation energy is

$$G_{\text{pol}} = \frac{1}{2}\int \phi_{\text{reaction}}(\mathbf{r})\rho(\mathbf{r})\ \mathrm{d}\mathbf{r}.$$

For the ion, the charge density is $\rho(\mathbf{r}) = q\delta_0(\mathbf{r}-\mathbf{c})$, so the electrostatic solvation energy is

$$G_{\text{pol}} = -\frac{q^2}{2a}(1-\frac{1}{\varepsilon}). \tag{3.2.4}$$

Equation (3.2.4) is also known as the Born formula [49].

For the case of multiple ions, e.g. a molecule consisting of M atoms where atom i $(i = 1, \ldots, M)$ is represented as a ball of radius a_i centered at $\mathbf{x}_i$ and contains a point charge q_i at the center, if the distances between any pair of atoms are sufficiently large compared with their radii, the electrostatic solvation energy is simply the sum of the individual Born terms plus the effect of the dielectric medium on the pairwise Coulomb interactions of the charges in the molecule:

$$G_{\text{pol}} = \sum_i -\frac{q_i^2}{2a_i}(1-\frac{1}{\varepsilon}) + \sum_i \sum_{j\neq i} -\frac{q_i q_j}{2r_{ij}}(1-\frac{1}{\varepsilon}). \tag{3.2.5}$$

In a real molecule the atoms could be very close to each other, for example two atoms highly overlap if they are connected by a strong chemical bond. In (3.2.5) some of the low dielectric overlapping regions are mistaken as high dielectric, which leads to a larger energy difference after the solvation.

Hence (3.2.5) overestimates the magnitude of the electrostatic solvation energy of the molecule case.

The best and most often used generalized Born (GB) function for electrostatic solvation energy is Still's formula [46]:

$$G_{\text{pol}} = -\frac{1}{2}(1 - \frac{1}{\varepsilon}) \sum_{i,j} \frac{q_i q_j}{[r_{ij}^2 + R_i R_j \exp(-\frac{r_{ij}^2}{4R_i R_j})]^{\frac{1}{2}}} \tag{3.2.6}$$

where R_i is the effective Born radius of atom i and r_{ij} is the distance between atom i and atom j.

To examine the GB function, we build a model of two ions immersed in a dielectric medium with $\varepsilon = 80$. Each ion is modeled as a sphere of radius $a_i = 2$ Å with point charge $q_i = 1\text{e}$ where e is the charge of an electron ($\approx 1.602 \times 10^{-19}$ coulombs). We vary the distance r_{12} between the two ions from 6 Å to 0.2 Å. For this simple model we can analytically compute G_{pol}:

$$G_{\text{pol}} = \begin{cases} -(1 - \frac{1}{\varepsilon}) \left(\frac{q_1^2}{2a_1} + \frac{q_1 q_2}{2a_2} \right) - (1 - \frac{1}{\varepsilon}) \left(\frac{q_1 q_2}{2a_1} + \frac{q_2^2}{2a_2} \right), & r_{12} < 2; \\ -(1 - \frac{1}{\varepsilon}) \left(\frac{q_1^2}{2a_1} + \frac{q_1 q_2}{2r_{12}} \right) - (1 - \frac{1}{\varepsilon}) \left(\frac{q_1 q_2}{2r_{12}} + \frac{q_2^2}{2a_2} \right), & r_{12} \geq 2; \end{cases}$$

The analytical G_{pol} is plotted as the red curve in Figure 3.1. For the same model we compute G_{pol} by using (3.2.5) and (3.2.6) and plot the results in Figure 3.1 as the blue curve and the red curve, respectively. We see that as the ions get closer, the error of the G_{pol} computed from (3.2.5) becomes large, while the G_{pol} computed from Still's GB is acceptable.

In [50], salt effects are incorporated in the GB model:

$$G_{\text{pol}} = -\frac{1}{2} \sum_{i,j} (1 - \frac{e^{-\kappa f_{\text{GB}}}}{\varepsilon}) \frac{q_i q_j}{f_{\text{GB}}}, \tag{3.2.7}$$

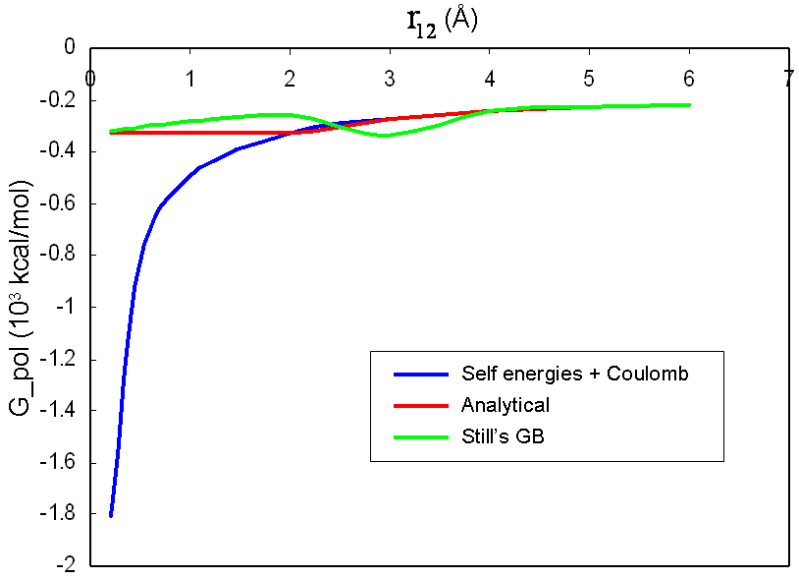


Figure 3.1: Electrostatic solvation energy for two equal charges of equal radii with different separation. Red curve: the analytical solution. Green curve: the Still's GB approximation. Blue curve: Born self energies + the screened Coulomb interaction.

where $f_{\mathrm{GB}} = [r_{ij}^2 + R_i R_j \exp(-\frac{r_{ij}^2}{4R_iR_j})]^{\frac{1}{2}}$, κ^{-1} is the Debye screening length given by

$$\kappa^{-1} = \sqrt{\frac{\varepsilon k_B T}{2I}}. \tag{3.2.8}$$

Here k_B is the Boltzmann constant, T is the absolute temperature, and I is the ionic strength:

$$I = \frac{1}{2}\sum_i n_i q_i^2 \tag{3.2.9}$$

with the summation over different species of ions, q_i being the ionic charge, and n_i being the ionic concentration in number of ions of species i per volume.

In the GB model, one needs to compute the effective Born radius R_i for every atom. After the Born radii are computed, one can compute (3.2.6) by

the fast multipole method (FMM) [51] with the complexity being $O(M \log M)$ where M is the number of atoms in a molecule. The Born radii computation is not trivial and is the most time-consuming part of the overall computation. In the next section we are going to discuss how to efficiently do this computation.

3.3 Computation of the effective Born radii

3.3.1 Effective Born radius

The effective Born radius of an atom reflects the depth at which the atom is buried inside the molecule. For an atom buried deep in a molecule it has a larger Born radius, whereas for an atom close to the surface it has a smaller radius as shown in Figure 3.2. Hence in the GB model (3.2.6) the surfactant atoms have a stronger impact on the polarization than the inner atoms.

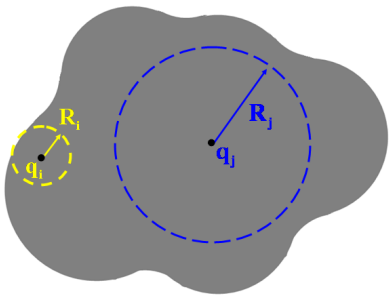


Figure 3.2: The effective Born radius reflects the depth a charge is buried inside the molecule. The Born radius of an atom is small if the atom is close to the surface of the molecule, otherwise the Born radius is large and therefore has weaker interactions with the solvent.

To compute the Born radius R_i, one may temporarily assume that only

atom i has a charge q_i and all the other atoms in the molecule have zero charges. Then from (3.2.6),

$$G_{\text{pol}} = -\frac{1}{2}(1 - \frac{1}{\varepsilon})\frac{q_i^2}{R_i}. \tag{3.3.1}$$

For the same model, one can also compute G_{pol} from (3.1.3), then

$$G_{\text{pol}} = \frac{1}{2}q_i(\phi_{\text{solvent}}(\mathbf{x}_i) - \phi_{\text{gas-phase}}(\mathbf{x}_i)) \tag{3.3.2}$$

where $\mathbf{x}_i$ is the center of atom i. Combining (3.3.1) and (3.3.2), one can solve for R_i. In this method, even though one can 'accurately' compute R_i once one knows the electric potentials involved in (3.3.2), however one has to solve the Poisson equations twice. Hence this procedure would have no practical advantage over the PB method.

The other way of computing the Born radii, as derived in [52], is to write down the electric potential energy in terms of the dot product of the electric field $\mathbf{E}$ and the electric displacement $\mathbf{D}$:

$$E^{\text{ele}} = \frac{1}{2}\int \rho(\mathbf{r})\phi(\mathbf{r})\ \mathrm{d}\mathbf{r} \tag{3.3.3}$$

$$= \frac{1}{8\pi}\int \mathbf{E}(\mathbf{r}) \cdot \mathbf{D}(\mathbf{r})\ \mathrm{d}\mathbf{r} \tag{3.3.4}$$

This is because from the Poisson equation, we have

$$\rho(\mathbf{r}) = -\frac{1}{4\pi}\nabla \cdot (\varepsilon(\mathbf{r})\nabla\phi(\mathbf{r}))\,. \tag{3.3.5}$$

By substituting (3.3.5) into (3.3.3), integrating by parts, and noticing that $\mathbf{E} = -\nabla\phi$, $\mathbf{D} = \varepsilon\mathbf{E}$, and $\phi(\mathbf{r}) \downarrow 0$ as $\mathbf{r} \uparrow \infty$, we get (3.3.4), which can be

rewritten as

$$E^{\text{ele}} = \frac{1}{8\pi} \int (\mathbf{D}/\varepsilon) \cdot \mathbf{D} \, \mathrm{d}\mathbf{r}. \tag{3.3.6}$$

Assuming that the electric displacement $\mathbf{D}$ is approximated as the Coulomb field, then the displacement due to the charge of atom i is

$$\mathbf{D} \approx \frac{q_i(\mathbf{r} - \mathbf{x}_i)}{|\mathbf{r} - \mathbf{x}_i|^3}.$$

This is known as the Coulomb field approximation which is an essential approximation used in the GB theory. Then for the same model where there is a single charge q_i in the molecule, the electrostatic energy of the system becomes

$$E^{\text{ele}} = \frac{1}{8\pi} \int_{\text{in}} \frac{q_i^2}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r} + \frac{1}{8\pi} \int_{\text{ex}} \frac{q_i^2}{\varepsilon |\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r}.$$

Computing the difference of the electrostatic energy when the medium dielectric constant changes from 1.0 to ε, we get the electrostatic solvation energy

$$G_{\text{pol}} = -\frac{1}{8\pi} \left(1 - \frac{1}{\varepsilon}\right) \int_{\text{ex}} \frac{q_i^2}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r}. \tag{3.3.7}$$

By setting (3.3.7) and (3.3.1) equal, one can solve for the Born radius:

$$R_i^{-1} = \frac{1}{4\pi} \int_{\text{ex}} \frac{1}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r}. \tag{3.3.8}$$

There are various ways of computing the Born radius based on (3.3.8). These methods can be divided into two categories [2]:

- Volume integration methods (V-GB) [20, 22, 46, 53–55] which compute R_i by either directly using (3.3.8) or using the equivalent form

$$R_i^{-1} = a_i^{-1} - \frac{1}{4\pi} \int_{\text{in}, |\mathbf{r} - \mathbf{x}_i| > a_i} \frac{1}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r}; \tag{3.3.9}$$

- Surface integration methods (S-GB) [53] which converts the volume integration in (3.3.8) to a surface integration by the divergence theorem

$$\int_V \nabla \cdot \mathbf{f} \, \mathrm{d}\mathbf{r} = \int_S \mathbf{f} \cdot \mathbf{n} \, \mathrm{d}S$$

with $\mathbf{f} = -\frac{\mathbf{r}-\mathbf{x}_i}{|\mathbf{r}-\mathbf{x}_i|^4}$, which yields

$$R_i^{-1} = \frac{1}{4\pi} \int_\Gamma \frac{(\mathbf{r}-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4} \, \mathrm{d}S, \tag{3.3.10}$$

where Γ is the boundary of the molecule and $\mathbf{n}(\mathbf{r})$ is the normal of the molecular surface at $\mathbf{r}$ pointing to the solvent.

In general, the surface integration methods are more efficient than the volume integration methods due to the decreased dimension. For this reason, in our work we utilize the surface integration method to compute the Born radii and we model Γ by using the ASMS introduced in Chapter 2.

3.3.2 Correction of the Born radius calculation

Because the Coulomb field approximation treats each atom in an isolated state, which neglects the reaction field among the atoms, the self-solvation energy is underestimated and consequently the Born radii are overestimated. Empirical correction terms have been developed to compensate the error caused by the Coulomb field approximation [20, 22, 53, 54]. In [20, 54] a higher order correction term is added:

$$R_i^{-1} = (1 - \frac{1}{\sqrt{2}})(R_i^0)^{-1} + \mathrm{corr}, \tag{3.3.11}$$

where R_i^0 is the initial Born radius calculated from (3.3.8) or (3.3.10) and

$$\text{corr} = \left(\frac{1}{4a_i^4} - \frac{1}{4\pi} \int_{\text{in},|\mathbf{r}-\mathbf{x}_i|>a_i} \frac{1}{|\mathbf{r}-\mathbf{x}_i|^7} \, \mathrm{d}\mathbf{r} \right)^{1/4}. \tag{3.3.12}$$

The correction term (3.3.12) is equivalent to

$$\text{corr} = \left(\frac{1}{4\pi} \int_{\text{ex}} \frac{1}{|\mathbf{r}-\mathbf{x}_i|^7} \, \mathrm{d}\mathbf{r} \right)^{1/4}. \tag{3.3.13}$$

We apply the divergence theorem once again to convert (3.3.13) to a surface integration:

$$\text{corr} = \left(\frac{1}{16\pi} \int_{\Gamma} \frac{(\mathbf{r}-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^7} \, \mathrm{d}\mathbf{r} \right)^{1/4}. \tag{3.3.14}$$

3.4 NFFT-based GB computation

3.4.1 Geometric model of Γ

The Born radii computations are based on integrals over the molecular surface Γ. We adopt the methods introduced in Chapter 2 to generate the geometric model of Γ.

This mesh is excessively detailed and therefore is expensive if it is used in the Born radii computation. In Chapter 2 we conclude that the ASMS requires only a coarse mesh to get a good approximation. So we simplify the triangular mesh by using the quadric-based simplification algorithm [38] and apply the geometric flow technique to improve the quality of the mesh [35]. We finally build the ASMS model based on the decimated and regularized mesh (Figure 3.3).

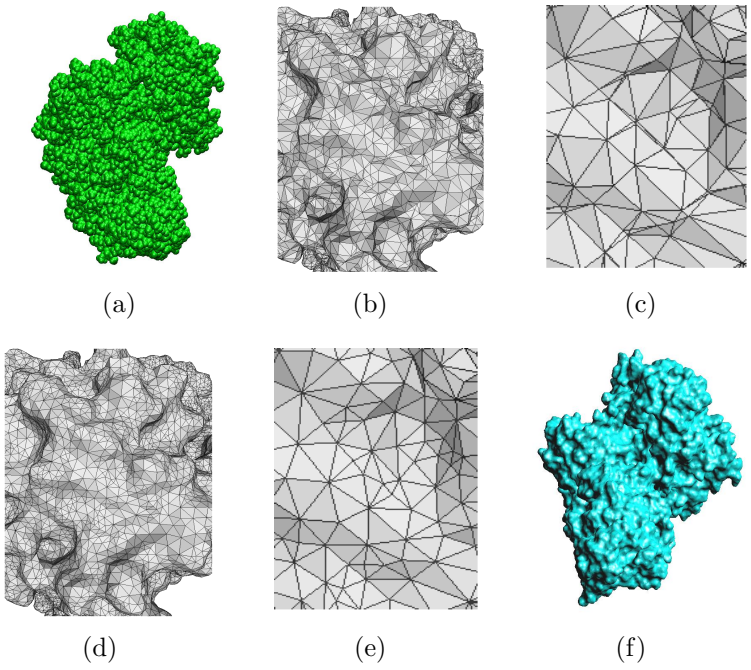


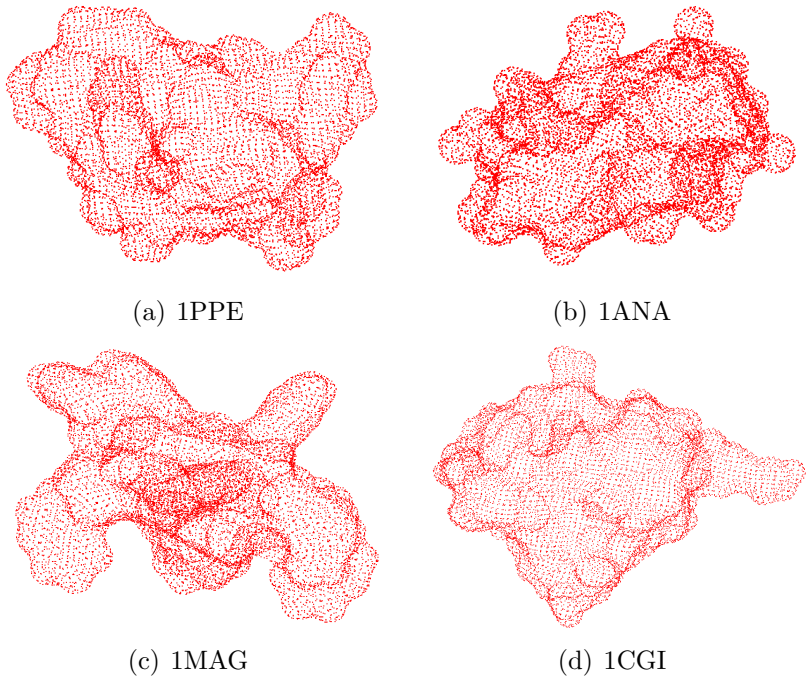
Figure 3.3: (a) Discrete van der Waals model of protein 1BGX with 19,647 atoms; (b) and (c) zoom-in views of the the initial triangulation of the continuum surface with 85656 triangles; (d) and (e) zoom-in views of the quality improved mesh; (f) continuum ASMS model generated based on the quality improved mesh.

3.4.2 Fast radius calculation

The molecular surface Γ is represented as the union of a set of algebraic patches. We numerically evaluate (3.3.10) by following the method discussed in Section 2.4.1, where the Gaussian quadrature rule is applied on each patch. The overall discretization becomes

$$R_i^{-1} = \frac{1}{4\pi} \sum_{k=1}^{N} w_k \frac{(\mathbf{r}_k - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4} \qquad i = 1, \ldots, M \tag{3.4.1}$$

where w_k and $\mathbf{r}_k$ are the Gaussian integration weights and nodes on Γ (Figure 3.4).



(a) 1PPE (b) 1ANA

(c) 1MAG (d) 1CGI

Figure 3.4: Gaussian integration points on the surface of (a) 1PPE, (b) 1ANA, (c) 1MAG, and (d) 1CGI.

In the AMSM generation introduced in Section 2.2.3, (2.22) defines a transformation $\mathcal{J} : (b_1, b_2, \lambda) \rightarrow (x, y, z)$. The nodes $\mathbf{r}_k$ are computed by mapping the Gaussian nodes of the base triangle to the curved patch via the transformation $\mathcal{J}$. Suppose $\mathbf{r}_k^0$ and w_k^0 are the Gaussian quadrature nodes and weights on the base triangle (Table 3.1), then $\mathbf{r}_k = \mathcal{J}(\mathbf{r}_k^0)$ and $w_k = w_k^0 \sqrt{EG - F^2}|_{\mathbf{r}_k^0}$ where E, F, and G are defined in Section 2.4.1.

n	W_i	b_1^i	b_2^i, b_3^i	m
1	1.000000000000000	0.333333333333333	0.333333333333333	
			0.333333333333333	1
3	0.333333333333333	0.666666666666667	0.166666666666667	
			0.166666666666667	3
4	-0.562500000000000	0.333333333333333	0.333333333333333	
			0.333333333333333	1
	0.520833333333333	0.600000000000000	0.200000000000000	
			0.200000000000000	3

Table 3.1: One-point, three-point, and four-point Gaussian quadrature rules of a triangle [1]. (b_1^i, b_2^i, b_3^i) are the barycentric coordinates of the ith evaluation point, W_i are the weights, m indicates the number of permutations of the corresponding point.

If we evaluate (3.4.1) in the trivial way, the complexity is $O(MN)$. Instead of doing that, we adopt the NFFT-based fast summation algorithm which improves the complexity of evaluating summations of the form

$$G(\mathbf{x}_i) = \sum_{k=1}^{N} c_k g(\mathbf{x}_i - \mathbf{r}_k), \qquad i = 1, \ldots, M,$$

to $O(M + N + n^3 \log n)$ [56]. The details of this algorithm along with the NFFT algorithm are discussed in Appendix B.

In order to apply the fast summation algorithm, we split (3.4.1) into

$$R_i^{-1} = \frac{1}{4\pi} \sum_{k=1}^{N} \frac{w_k \mathbf{r}_k \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4} - \frac{1}{4\pi} \sum_{k=1}^{N} \frac{w_k \mathbf{x}_i \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4} \tag{3.4.2}$$

We split again the second summation in (3.4.2):

$$\sum_{k=1}^{N} \frac{w_k \mathbf{x}_i \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4} = \sum_{k=1}^{N} \frac{w_k x_i n_x^k}{|\mathbf{r}_k - \mathbf{x}_i|^4} + \sum_{k=1}^{N} \frac{w_k y_i n_y^k}{|\mathbf{r}_k - \mathbf{x}_i|^4} + \sum_{k=1}^{N} \frac{w_k z_i n_z^k}{|\mathbf{r}_k - \mathbf{x}_i|^4}. \tag{3.4.3}$$

Thus the first summation in (3.4.2) and the three summations in (3.4.3) are well formalized. We may easily apply the fast summation method to each of them by setting the kernel function as $g(\mathbf{x} - \mathbf{x}_k) = \frac{1}{|\mathbf{r}_k - \mathbf{x}|^4}$ and the coefficient $c_k = w_k \mathbf{r}_k \cdot \mathbf{n}(\mathbf{r}_k)$, $w_k x_i n_x^k$, $w_k y_i n_y^k$, $w_k z_i n_z^k$, respectively. Since all the integration nodes $\mathbf{r}_k$ are on the molecular surface, there is no singularity in the kernel function.

For a molecule of M atoms, the overall time complexity of computing G_{pol} which involves computing the N quadrature points, the M Born radii, and the summation of (3.2.6) is $O(M + N + n^3 \log n)$, where n is a parameter used in the fast summation algorithm.

3.4.3 Error analysis

In the whole computation procedure, we introduce a quadrature error E_{Q} when discretizing (3.3.10) into (3.4.1), a fast summation error E_{FS} when truncating the Fourier series (B.1.2) into finite terms, NFFT^{T} errors $E_{\boldsymbol{\omega}}$ when

computing the coefficients (B.1.6) in the fast summation algorithm, and an NFFT error E_{NFFT} when finally evaluating (B.1.5) by the NFFT algorithm.

Let I_i denote the exact integration of (3.3.10) for atom i, and $\tilde{I}_i$ denote the final output of the numerical computation. We have

$$I_i = \tilde{I}_i + E_{\mathrm{NFFT}} + E_{\mathrm{NFFT}^{\mathrm{T}}} + E_{\mathrm{FS}} + E_{\mathrm{Q}}.$$

Let $||E||_\infty = \max_i |I_i - \tilde{I}_i|$, then we have

$$||E||_\infty \leq ||E_{\mathrm{Q}}||_\infty + ||E_{\mathrm{FS}}||_\infty + ||E_{\mathrm{NFFT}}||_\infty + ||E_{\mathrm{NFFT}^{\mathrm{T}}}||_\infty. \tag{3.4.4}$$

Next we are going to analyze each individual error $||E_{\mathrm{Q}}||_\infty$, $||E_{\mathrm{FS}}||_\infty$, $||E_{\mathrm{NFFT}}||_\infty$, and $||E_{\mathrm{NFFT}^{\mathrm{T}}}||_\infty$ in the next few sections.

3.4.3.1 Quadrature error

Let Γ_e be one of the algebraic patches on the molecular surface Γ. Suppose Γ_e is built based on a triangle $e := [\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k]$. Any point $(b_1, b_2, b_3) \in e$ can be mapped to a point $\mathbf{r}(b_1, b_2) \in \Gamma_e$. The integration (3.3.10) over Γ_e is

$$\begin{aligned} I_e &= \int_{\Gamma_e} \frac{(\mathbf{r} - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}S \\ &= \iint_{\Omega_0} \frac{(\mathbf{r}(b_1, b_2) - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r}(b_1, b_2))}{|\mathbf{r}(b_1, b_2) - \mathbf{x}_i|^4} \, |J| \, \mathrm{d}b_1 \mathrm{d}b_2 \end{aligned} \tag{3.4.5}$$

where Ω_0 is the canonical triangle, (b_1, b_2, b_3) is the barycentric coordinates of the points in Ω_0 and $|J|$ is the Jacobian. Let $f(b_1, b_2)$ denote the integrand in (3.4.5), then

$$I_e = \iint_{\Omega_0} f(b_1, b_2) \, \mathrm{d}b_1 \, \mathrm{d}b_2 = \sum_{k=1}^{n} w_k f(b_1^k, b_2^k) + E. \tag{3.4.6}$$

As we show in Appendix C, $f(b_1, b_2) \in C^\infty(\Omega_0)$. Suppose we use an n-th order quadrature rule. We expand $f(b_1, b_2)$ in a Taylor series around a point $(b_1', b_2', b_3') \in \Omega_0$:

$$f(b_1, b_2) = P_n(b_1, b_2) + R_n(b_1, b_2) \tag{3.4.7}$$

where $P_n(b_1, b_2)$ is a polynomial of degree n:

$$P_n(b_1, b_2) = f(b_1', b_2') + \frac{1}{n!}[(b_1 - b_1')\frac{\partial}{\partial b_1} + (b_2 - b_2')\frac{\partial}{\partial b_2}]^n f(b_1', b_2') \tag{3.4.8}$$

and the residue R_n is

$$R_n(b_1, b_2) = \frac{1}{(n+1)!}[(b_1 - b_1')\frac{\partial}{\partial b_1} + (b_2 - b_2')\frac{\partial}{\partial b_2}]^{n+1} f(b_1^*, b_2^*), \qquad (b_1^*, b_2^*) \in \Omega_0. \tag{3.4.9}$$

Then the error E becomes

$$E = \iint_{\Omega_0} R_n(b_1, b_2)\, \mathrm{d}b_1\, \mathrm{d}b_2 - \sum_{k=1}^{3} w_k R_n(b_1^k, b_2^k).$$

Let $W_k = \max(|w_k|)$, we get

$$|E| \le \iint_{\Omega_0} |R_n(b_1, b_2)|\, \mathrm{d}b_1\, \mathrm{d}b_2 + W_k \sum_{k=1}^{3} |R_n(b_1^k, b_2^k)|.$$

Within Ω_0, $|b_1 - b_1'| \le 1$ and $|b_2 - b_2'| \le 1$, hence

$$|R_n(b_1, b_2)| \le \frac{1}{(n+1)!}[|\frac{\partial}{\partial b_1}| + |\frac{\partial}{\partial b_2}|]^{n+1} f(b_1^*, b_2^*), \tag{3.4.10}$$

where $|\frac{\partial}{\partial b}|\cdot$ denotes $|\frac{\partial \cdot}{\partial b}|$. By the chain rule,

$$\frac{\partial}{\partial b_1} = \frac{\partial}{\partial x}\frac{\partial x}{\partial b_1} + \frac{\partial}{\partial y}\frac{\partial y}{\partial b_1} + \frac{\partial}{\partial z}\frac{\partial z}{\partial b_1}.$$

According to (2.22), we have $\frac{\partial x}{\partial b_1} = v_1^x - v_3^x + \lambda(n_1^x - n_3^x)$. Let h_{max} be the maximum edge length of the triangular mesh, $\lambda_{max} = \max\{|\lambda|\}$, and $h = \max(h_{max}, \lambda_{max})$. Then we have $|\frac{\partial x}{\partial b_1}| \leq 2h$. Similarly, we can get the same bound for the derivatives of x, y, z with respect to b_1 and b_2. Therefore

$$|R_n(b_1, b_2)| \leq \frac{(2h)^{n+1}}{(n+1)!}[|\frac{\partial}{\partial x}| + |\frac{\partial}{\partial y}| + |\frac{\partial}{\partial z}|]^{n+1}\tilde{f}(x^*, y^*, z^*) \leq C\frac{(2h)^{n+1}}{(n+1)!} \tag{3.4.11}$$

where $(x^*, y^*, z^*) = b_1^*\mathbf{v}_i(\lambda) + b_2^*\mathbf{v}_j(\lambda) + b_3^*\mathbf{v}_k(\lambda)$, $\tilde{f}(x^*, y^*, z^*) = f(b_1^*, b_2^*)$, and the constant $C = \max\limits_{(x,y,z)\in\Gamma} |D^{n+1}\tilde{f}(x, y, z)| < \infty$. Noticing that the area of Ω_0 is $1/2$, we can write

$$|E| \leq (\frac{1}{2} + W_k)\frac{2^{n+1}}{(n+1)!}Ch^{n+1}. \tag{3.4.12}$$

Even though a greater number of quadrature nodes corresponds with the higher order accuracy, the increase in complexity is a limiting factor. As a trade off, we use a two dimensional 3-point Gaussian quadrature over the triangle Ω_0 which is of order 2 [1]. The nodes are $(\frac{1}{6}, \frac{1}{6}, \frac{1}{3})$ and its permutations. The weight associated with each node is $\frac{1}{3}$. Then

$$|E| \leq \frac{10}{9}Ch^3. \tag{3.4.13}$$

Suppose there are N_e patches on Γ, then $|E_{\mathrm{Q}}| \leq \frac{10}{9}N_eCh^3$. So we have the same bound

$$||E_{\mathrm{Q}}||_\infty \leq \frac{10}{9}N_eCh^3. \tag{3.4.14}$$

3.4.3.2 Fast summation error

According to the fast summation method described in Appendix B.1, the Fourier series is truncated into a finite number of terms

$$E_{\mathrm{FS}} := \sum_{k=1}^{N} c_k \left(\sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} g_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega} \cdot (\mathbf{x}_i - \mathbf{r}_k)} \right) = \sum_{k=1}^{N} c_k T_k$$

where T_k denotes the truncation error of the Fourier series. By Hölder's inequality

$$|E_{\mathrm{FS}}| \leq ||\mathbf{c}||_\infty \sum_{k=1}^{N} |T_k| \tag{3.4.15}$$

where $||\mathbf{c}||_\infty := \max_{k=1,\ldots,N} |c_k|$,

$$|T_k| = | \sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} g_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega} \cdot (\mathbf{x}_i - \mathbf{r}_k)} | \leq \sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} |g_{\boldsymbol{\omega}}| \tag{3.4.16}$$

with

$$g_{\boldsymbol{\omega}} = \int_\Pi g(\mathbf{x}) e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{x}} \, \mathrm{d}\mathbf{x} \tag{3.4.17}$$

and g being the the kernel function in the fast summation. In the Born radii calculation, $g(\mathbf{x}) = \frac{1}{|\mathbf{x}|^4}$. As defined in Appendix B.1, Π is bounded and excludes 0. Let $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$. Then we rewrite $\sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} |g_{\boldsymbol{\omega}}|$ as

$$\sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} |g_{\boldsymbol{\omega}}| = \sum_{i,j,k=0,1} \sum_{\omega_1=n+1}^{\infty} \sum_{\omega_2=n+1}^{\infty} \sum_{\omega_3=n+1}^{\infty} |g_{(-1)^i \omega_1 \, (-1)^j \omega_2 \, (-1)^k \omega_3}|. \tag{3.4.18}$$

By successive integration by parts for each dimension, we get

$$g_{\omega_1 \omega_2 \omega_3} = \left(\frac{-i}{2\pi\omega_1} \right)^{m_1} \left(\frac{-i}{2\pi\omega_2} \right)^{m_2} \left(\frac{-i}{2\pi\omega_3} \right)^{m_3} \int_\Pi D^m g(\mathbf{x}) e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{x}} \, \mathrm{d}\mathbf{x},$$

where $m = m_1 + m_2 + m_3$ and $D^m g = (\frac{\partial^{m_1}}{\partial x^{m_1}} + \frac{\partial^{m_2}}{\partial y^{m_2}} + \frac{\partial^{m_3}}{\partial z^{m_3}}) g$. Therefore

$$|g_{\omega_1 \omega_2 \omega_3}| \leq \frac{1}{(2\pi)^m \omega_1^{m_1} \omega_2^{m_2} \omega_3^{m_3}} \int_\Pi |D^m g(\mathbf{x})| \, \mathrm{d}\mathbf{x}.$$

Let $\mu_m = \int_\Pi |D^m g(\mathbf{x})| \, \mathrm{d}\mathbf{x}$. We obtain $|g_{\omega_1\omega_2\omega_3}| \leq \frac{\mu_m}{(2\pi)^m \omega_1^{m_1}\omega_2^{m_2}\omega_3^{m_3}}$. For the other terms in (3.4.18) we have the same upper bound. If we assume $m_1, m_2, m_3 \geq 2$, then

$$\begin{aligned}
|T_k| &\leq \frac{8\mu_m}{(2\pi)^m}\left(\sum_{\omega_1=n+1}^{\infty}\frac{1}{\omega_1^{m_1}}\right)\left(\sum_{\omega_2=n+1}^{\infty}\frac{1}{\omega_2^{m_2}}\right)\left(\sum_{\omega_3=n+1}^{\infty}\frac{1}{\omega_3^{m_3}}\right) \\
&\leq \frac{8\mu_m}{(2\pi)^m}\left(\int_n^{\infty}\frac{1}{\omega_1^{m_1}}\,\mathrm{d}\omega_1\right)\left(\int_n^{\infty}\frac{1}{\omega_2^{m_2}}\,\mathrm{d}\omega_2\right)\left(\int_n^{\infty}\frac{1}{\omega_3^{m_3}}\,\mathrm{d}\omega_3\right) \\
&= \frac{8\mu_m}{(2\pi)^m(m_1-1)(m_2-1)(m_3-1)n^{m-3}}.
\end{aligned}$$

For $m_1 = m_2 = m_3$, we have

$$|T_k| \leq \frac{8\mu_6}{(2\pi)^6 n^3}. \tag{3.4.19}$$

Then for (3.4.16), we have

$$|E_{\mathrm{FS}}| \leq ||\mathbf{c}||_\infty \frac{8\mu_6 N}{(2\pi)^6 n^3}. \tag{3.4.20}$$

We see (3.4.20) is independent of i, and therefore we get

$$||E_{\mathrm{FS}}||_\infty \leq ||\mathbf{c}||_\infty \frac{8\mu_6 N}{(2\pi)^6 n^3}. \tag{3.4.21}$$

3.4.3.3 NFFT error

When computing (B.1.1) by using the NFFT algorithm, two steps of approximations are introduced, which raise the *aliasing error* E^1_{NFFT} and the *truncation error* E^2_{NFFT} [57]. So

$$||E_{\mathrm{NFFT}}||_\infty \leq ||E^1_{\mathrm{NFFT}}||_\infty + ||E^2_{\mathrm{NFFT}}||_\infty.$$

The error bounds of E^1_{NFFT} and E^2_{NFFT} are

$$||E^1_{\mathrm{NFFT}}||_\infty \leq ||\hat{G}||_1 \max_{\boldsymbol{\omega}\in I_n} \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} |\frac{C_{\boldsymbol{\omega}+\mathbf{i}\sigma n}(\xi)}{C_{\boldsymbol{\omega}}(\xi)}|, \tag{3.4.22}$$

$$||E^2_{\mathrm{NFFT}}||_\infty \leq \frac{1}{\sigma^3 n^3} \max_{\boldsymbol{\omega}\in I_n}(C^{-1}_{\boldsymbol{\omega}}(\xi))||\hat{G}||_1 \max_i \sum_{\mathbf{l}\in I_{\sigma n}} |\xi(\mathbf{x}_i - \frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}_i - \frac{\mathbf{l}}{\sigma n})|, \tag{3.4.23}$$

where ξ is a 1-periodic function defined in Appendix B.2, $C_{\boldsymbol{\omega}}(\xi)$ are the Fourier coefficients of ξ, and η is a truncated version of ξ. In the fast summation algorithm (B.1.5), $||\hat{G}||_1 = \sum_{\boldsymbol{\omega}\in I_n} |g_{\boldsymbol{\omega}} a_{\boldsymbol{\omega}}|$, where $g_{\boldsymbol{\omega}}$ and $a_{\boldsymbol{\omega}}$ are defined in Appendix B.1. Combining (3.4.22) and (3.4.23), one obtains

$$||E_{\mathrm{NFFT}}||_\infty \leq C(\xi, m, \sigma)||\hat{G}||_1. \tag{3.4.24}$$

$C(\xi, m, \sigma)$ are given in [56] for ξ being the

- Gaussian: $C(\xi, m, \sigma) = 4e^{-m\pi(1-1/(2\sigma-1))}$,
- cardinal central B-splines: $C(\xi, m, \sigma) = 4(\frac{1}{2\sigma-1})^{2m}$,
- powers of sinc function: $C(\xi, m, \sigma) = \frac{1}{m-1}\left(\frac{2}{\sigma^{2m}} + (\frac{\sigma}{2\sigma-1})^{2m}\right)$,
- Kaiser-Bessel function: $C(\xi, m, \sigma) = 5\pi^2 m^{3/2}\sqrt[4]{1-\frac{1}{\sigma}}e^{-m2\pi\sqrt{1-1/\sigma}}$.

3.4.4 NFFT$^{\mathrm{T}}$ error

$$E_{\mathrm{NFFT}^{\mathrm{T}}} = \sum_{\boldsymbol{\omega}\in I_n} g_{\boldsymbol{\omega}} E_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega}\cdot\mathbf{x}_i}, \tag{3.4.25}$$

where $E_{\boldsymbol{\omega}}$ denotes the error of the NFFT$^{\mathrm{T}}$ algorithm and $g_{\boldsymbol{\omega}}$ is the same as is defined in (B.2.6). Then we have

$$|E_{\mathrm{NFFT}^{\mathrm{T}}}| \leq \sum_{\boldsymbol{\omega}\in I_n} |g_{\boldsymbol{\omega}} E_{\boldsymbol{\omega}}| \leq \max_{\boldsymbol{\omega}\in I_n} |E_{\boldsymbol{\omega}}| \sum_{\boldsymbol{\omega}\in I_n} |g_{\boldsymbol{\omega}}| = ||E_{\boldsymbol{\omega}}||_\infty \, ||g||_1 \tag{3.4.26}$$

with $||E_{\boldsymbol{\omega}}||_\infty := \max_{\boldsymbol{\omega}\in I_n} |E_{\boldsymbol{\omega}}|$ and $||g||_1 := \sum_{\boldsymbol{\omega}\in I_n} |g_{\boldsymbol{\omega}}|$.

The NFFT$^{\mathrm{T}}$ error $E_{\boldsymbol{\omega}}$ is composed of the aliasing error ($E^1_{\boldsymbol{\omega}}$) and the truncation error ($E^2_{\boldsymbol{\omega}}$), $E_{\boldsymbol{\omega}} = E^1_{\boldsymbol{\omega}} + E^2_{\boldsymbol{\omega}}$. Hence

$$||E_{\boldsymbol{\omega}}||_\infty \leq ||E^1_{\boldsymbol{\omega}}||_\infty + ||E^2_{\boldsymbol{\omega}}||_\infty,$$

where $||E^1_{\boldsymbol{\omega}}||_\infty = \max_{\boldsymbol{\omega}\in I_n} |E^1_{\boldsymbol{\omega}}|$ and $||E^2_{\boldsymbol{\omega}}||_\infty = \max_{\boldsymbol{\omega}\in I_n} |E^2_{\boldsymbol{\omega}}|$. Based on the error bounds derived in Appendix B.3,

$$||E^1_{\boldsymbol{\omega}}||_\infty \leq ||\mathbf{c}||_1 \max_{\boldsymbol{\omega}\in I_n} \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} \frac{C_{\boldsymbol{\omega}+\mathbf{i}\sigma n}(\xi)}{C_{\boldsymbol{\omega}}(\xi)} \tag{3.4.27}$$

and

$$||E^2_{\boldsymbol{\omega}}||_\infty \leq \frac{1}{(\sigma n)^3} ||\mathbf{c}||_1 \max_{\boldsymbol{\omega}\in I_n} (C^{-1}_{\boldsymbol{\omega}}(\xi)) \max_k \sum_{\mathbf{l}\in I_{\sigma n}} |\xi(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k) - \eta(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k)| \tag{3.4.28}$$

where $||\mathbf{c}||_1 = \sum_{k=1}^N |c_k|$. Comparing (3.4.27) with (3.4.22) and comparing (3.4.28) with (3.4.23) yield the error estimation of $E_{\boldsymbol{\omega}}$ which is similar to E_{NFFT}:

$$||E_{\boldsymbol{\omega}}||_\infty \leq C(\xi, m, \sigma) ||\mathbf{c}||_1.$$

Hence

$$|E_{\mathrm{NFFT}^{\mathrm{T}}}| \leq C(\xi, m, \sigma) ||\mathbf{c}||_1 ||g||_1. \tag{3.4.29}$$

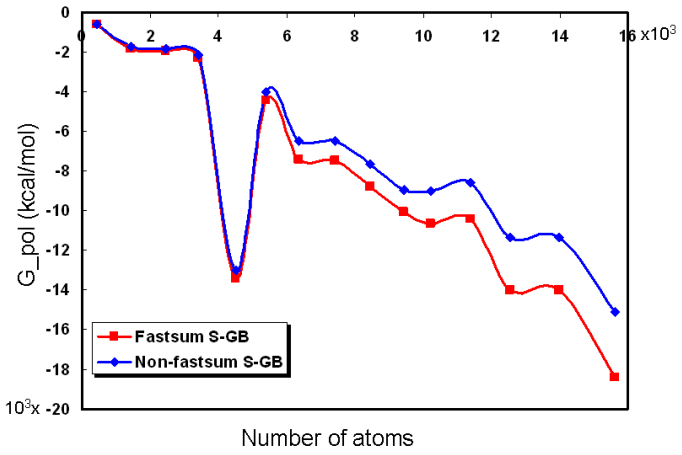
The inequality (3.4.29) is independent of i, therefore

$$||E_{\mathrm{NFFT}^{\mathrm{T}}}||_\infty \leq C(\xi, m, \sigma) ||\mathbf{c}||_1 ||g||_1. \tag{3.4.30}$$

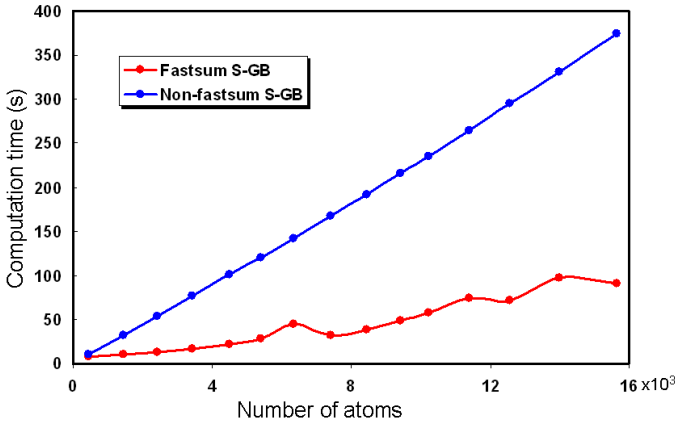
3.5 Results

As discussed in Section 3.4, the computational complexity and the error of the Born radii calculation are jointly controlled by the size of the molecule, the size of the triangular mesh, the parameter n introduced in the fast summation algorithm, and the parameters m and σ introduced in the NFFT algorithm. To investigate the impact of each factor, we do the following tests.

In Figure 3.5 we compare G_{pol} computed by the fast summation based S-GB and the trivial S-GB methods along with their computation time for proteins of various sizes. For all these proteins, we generate the ASMS of the same number of patches (in the practical test we use 20,000 patches for each protein). We choose the fixed parameters $n = 30$, $m = 4$, and $\sigma = 2$ for all the proteins. We observe that the G_{pol} computed by the fastsum S-GB is close to that computed by the trivial S-GB methods and the error gets larger as the molecule gets bigger. Even though in the error analysis in Section 3.4.3 it does not show that the error depends on the size of the molecule, the analysis is based on the assumption that the kernel function is defined on the domain $[-\frac{1}{2}, \frac{1}{2}]^3$. To ensure that $\mathbf{x}_i - \mathbf{r}_k$, $i = 1, \ldots, M$, $k = 1, \ldots, N$ are all within this range, we scale the molecule. The larger the molecule is the larger the scaling factor is. Later on when we scale back to the original coordinates by multiplying the scaling factor, the error gets amplified. As we expect, computation time of the fastsum S-GB increases as M becomes large but is much faster than the traditional S-GB method.



(a)

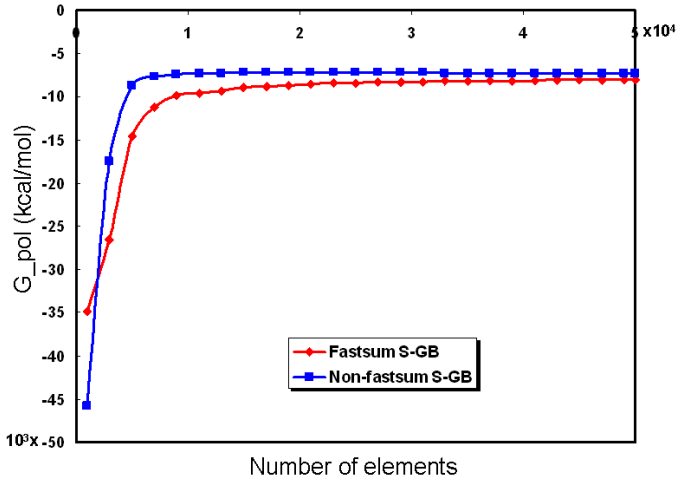


(b)

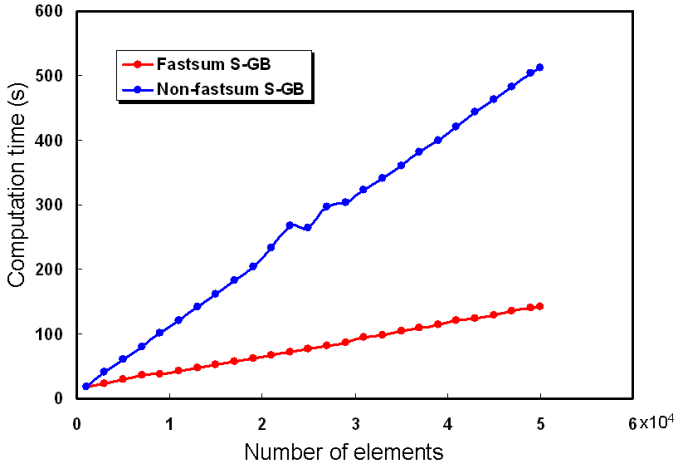
Figure 3.5: In (a) we compare G_{pol} computed by the fastsum S-GB and the non-fastsum S-GB for various proteins containing different number of atoms. In (b) we compare the computation time of the two methods.

In Figure 3.6, we compare G_{pol} computed by the fast summation based S-GB versus the trivial summation method and the computation time for a test protein 1JPS where we generate the ASMS with different numbers of patches. We use the same values of the parameters n, m, and σ as the previous test.

The result shows that as the mesh becomes finer and finer, the fastsum S-GB summation energy converges rapidly to the result of the trivial method and takes less computation time.



(a)



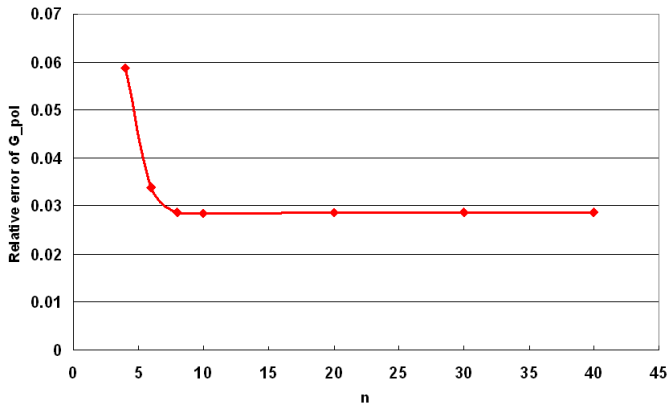
(b)

Figure 3.6: For protein 1JSP, in (a) we compare G_{pol} computed by the fastsum S-GB and the non-fastsum S-GB with various number of surface element. In (b) we compare the computation time of the two methods.

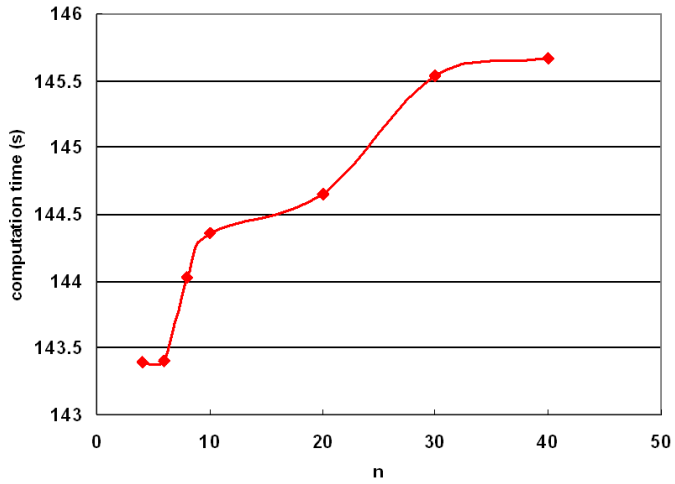
In Figure 3.7 we show the impact of the parameter n on the energy results and the computation time. For the test protein we generate an ASMS model with 50,000 triangular patches. $m = 4$ and $\sigma = 2$. The relative error is defined as $|G_{\text{pol}}^{\text{fast}} - G_{\text{pol}}^{\text{trivial}}/G_{\text{pol}}^{\text{trivial}}|$, where $G_{\text{pol}}^{\text{trivial}}$ is computed by the trivial S-GB method with the same ASMS model. The relative error converges as n increases and becomes stable for n greater than 10. Although the computation time of the fast S-GB increases as n gets larger, in comparison with the time of the trivial method, 511.96 seconds, it still much faster.

In the end, to examine the accuracy of the fastsum-SGB, we compare our results with two available packages Amber and DelPhi that are currently often used in MD simulations. Amber is a GB package where the Born radii are computed by using an empirical formula [58]. DelPhi is PB package where the PB equations are solved with finite difference methods. We test the G_{pol} calculation on a data set of 210 proteins. The size of the proteins varies from medium (about 400 atoms) to large (about 38,000 atoms). In Figure 3.8 we show the fastsum-SGB energy results (the y-axis) versus Amber 8.0 (the x-axis). The correlation coefficient is 0.9903. For those energies whose relative error is within 1%, we compare the computation time of fastsum-SGB and Amber. It turns out that in average Amber is 9.8 times faster than the fastsum-SGB. This is because Amber avoids computing the integrals of the Born radii by using an empirical formula. In Figure 3.9 we compare the fastsum-SGB energy results (the y-axis) versus DelPhi V.4 with grid spacing 0.5 Å. The correlation coefficient is 0.9812. Again for those energies whose

relative error is within 1%, we compare the computation time of fastsum-SGB and DelPhi. In average fastsum-SGB is 1.5 times faster than DelPhi.



(a)



(b)

Figure 3.7: For protein 1JSP, in (a) we show the relative error of G_{pol} computed by the fastsum S-GB to the non-fastsum S-GB as n varies. In (b) we show the computation time of the fastsum S-GB.

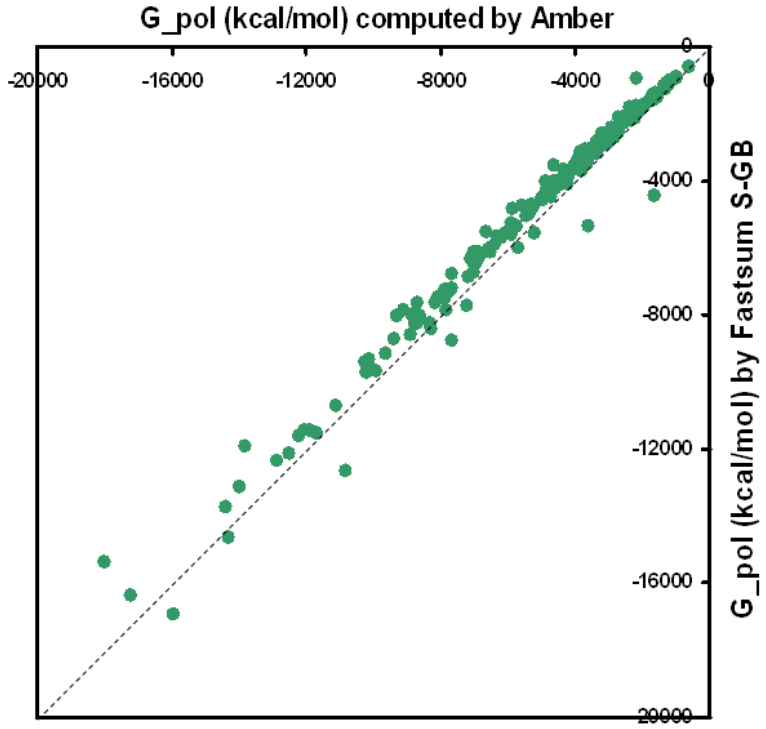


Figure 3.8: We test the $G_{\rm pol}$ calculation on a data set of 210 proteins from medium size (about 400 atoms) to large size (about 38,000 atoms). We compare the fastsum-SGB energy results (the y-axis) versus Amber 8.0 (the x-axis). The correlation coefficient is 0.9903.

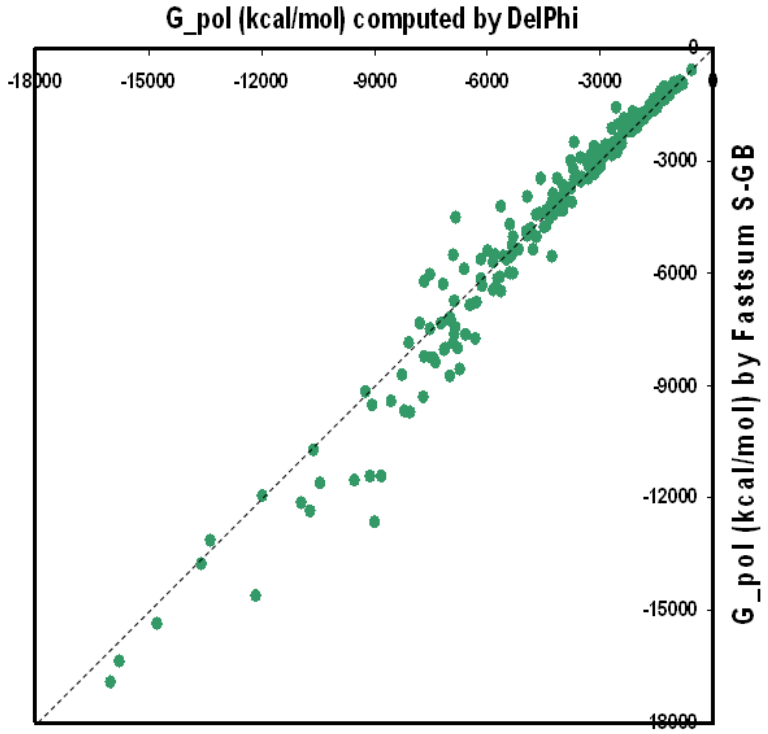


Figure 3.9: For the same test set as in Figure 3.8, we compare the fastsum-SGB energy results (the y-axis) versus DelPhi V.4 with grid spacing 5 Å. The correlation coefficient is 0.9812.

Chapter 4

Molecular Solvation Forces Computation

4.1 Introduction

The solvation force acting on the atoms is an important part of the forces that drive the molecular dynamics [59]. In general, if an atom has a strong solvation force, it indicates that the atom is sensitive to the change of the solvent environment, and hence is possible to be an active site of the molecule. This provides an easier and faster way than the experimental methods to detect the active sites of the proteins. On the other hand, for those atoms that have weak solvation forces, they can be tagged as stable atoms and may be updated less frequently than the active atoms during the MD simulations to save computation expense.

The solvation force on atom α is defined as

$$\mathbf{F}_{\alpha}^{\text{elec}} = -\frac{\partial \Delta G_{\text{sol}}}{\partial \mathbf{x}_{\alpha}}. \tag{4.1.1}$$

Similar to the energy function, one can partition the solvation force into the polar and the non-polar contributions:

$$\frac{\partial \Delta G_{\text{sol}}}{\partial \mathbf{x}_{\alpha}} = \frac{\partial}{\partial \mathbf{x}_{\alpha}}(G_{\text{cav}} + G_{\text{vdw}}) + \frac{\partial G_{\text{pol}}}{\partial \mathbf{x}_{\alpha}} = \gamma \frac{\partial A}{\partial \mathbf{x}_{\alpha}} + \frac{\partial G_{\text{pol}}}{\partial \mathbf{x}_{\alpha}}. \tag{4.1.2}$$

The non-polar force is proportional to the derivatives of the volume and/or the surface area with respect to the coordinates of the atoms. For large systems it is nontrivial to compute the gradients of the polar energy. The difficulty of computing the electrostatic force by using the PB method is that the sudden change of the dielectric constant across the molecular surface produces a discontinuous energy change and hence the force on the atoms on the boundary of the molecule becomes infinite [42]. Some attempts have been made to compute the electrostatic force in the GB method [54, 60].

In this chapter, we are going to explain how we extend the fast summation algorithm to the force computation in Section 4.2 and show some results in Section 4.3.

4.2 Fast Solvation Force Computation

To compute the polar force, we first define

$$G_{ij} = q_i q_j / (r_{ij}^2 + R_i R_j \exp(-\frac{r_{ij}^2}{4R_i R_j}))^{1/2}.$$

Then

$$G_{\text{pol}} = -\tau \sum_{i=1}^{M} \sum_{j=i+1}^{M} G_{ij} - \frac{\tau}{2} \sum_{i=1}^{M} G_{ii} \tag{4.2.1}$$

with $\tau = 1 - \frac{1}{\varepsilon}$. Differentiating (4.2.1) with respect to $\mathbf{x}_\alpha$, one gets

$$\frac{\partial G_{\text{pol}}}{\partial \mathbf{x}_\alpha} = -\tau \sum_{i=1}^{M} \sum_{j=i+1}^{M} \frac{\partial G_{ij}}{\partial \mathbf{x}_\alpha} - \frac{\tau}{2} \sum_{i=1}^{M} \frac{\partial G_{ii}}{\partial \mathbf{x}_\alpha}, \tag{4.2.2}$$

where

$$\frac{\partial G_{ij}}{\partial \mathbf{x}_\alpha} = \frac{\partial G_{ij}}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial \mathbf{x}_\alpha} + \frac{\partial G_{ij}}{\partial R_i} \frac{\partial R_i}{\partial \mathbf{x}_\alpha} + \frac{\partial G_{ij}}{\partial R_j} \frac{\partial R_j}{\partial \mathbf{x}_\alpha}. \tag{4.2.3}$$

The most difficult part in the force calculation is to compute the derivatives of the Born radii $\frac{\partial R_i}{\mathbf{x}_\alpha}$ for $i = 1, \ldots, M$. In (3.3.10), Γ is dependent on the positions of the atoms, so it is not easy to compute the derivative of R_i directly from (3.3.10). To solve this problem, we convert the integration domain back to the volume:

$$R_i^{-1} = \frac{1}{4\pi} \int_{\text{ex}} \frac{1}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r}. \tag{4.2.4}$$

By defining a volumetric density function to distinguish the exterior from the interior of the molecule, we may have an integration domain that is independent of $\{\mathbf{x}_i\}$. One way of defining the volumetric function is given in [21] where the density function for each of the atoms is defined as a characteristic function

$$\chi_i(\mathbf{r}) = \begin{cases} 1, & ||\mathbf{r} - \mathbf{x}_i|| \leq a_i \\ 0, & ||\mathbf{r} - \mathbf{x}_i|| > a_i \end{cases}$$

and the volumetric function is defined by following the inclusion-exclusion principle

$$\varrho(\mathbf{r}) = \sum_i \chi_i - \sum_{i<j} \chi_i \chi_j + \sum_{i<j<k} \chi_i \chi_j \chi_k - \sum_{i<j<k<l} \chi_i \chi_j \chi_k \chi_l + \ldots. \tag{4.2.5}$$

This model has some nice properties. For example, the exterior region of the molecule is well characterized by $\varrho = 0$. Two atoms i and j are disconnected if $\chi_i(\mathbf{r})\chi_j(\mathbf{r}) = 0$ for any $\mathbf{r} \in \mathbb{R}^3$. The drawback of this model is that function χ is not differentiable, which makes it not applicable to the derivative computation. Therefore we smoothen χ by introducing a cubic spline in a shell around each atom:

$$\varrho_i(\mathbf{r}) = \begin{cases} 1, & x \leq a_i \\ \frac{2}{w^3}(x - a_i)^3 - \frac{3}{w^2}(x - a_i)^2 + 1, & a_i < x < a_i + w \\ 0, & x \geq a_i + w \end{cases} \tag{4.2.6}$$

with $x = \| \mathbf{r} - \mathbf{x}_i \|$. The region is considered as the interior of atom i if $\rho_i(\mathbf{r}) \neq 0$. This region converges to the van der Waals volume of the atom as $w \downarrow 0$. In the SES model, if the distance between the centers of two atoms is greater than the sum of the radii of the atoms and the diameter of the probe, the two atoms are considered as completely separated. Otherwise they can always be connected by the reentrant surface of the rolling probe. In order to make the domain defined by $\varrho(\mathbf{r})$ have the same topology as the domain bounded by the SES, we set $w = 1.4$ Å. In addition, we ignore the cases that more than four atoms overlap simultaneously. Therefore the molecular volumetric density function becomes

$$\varrho(\mathbf{r}) = \sum_i \varrho_i - \sum_{i<j} \varrho_i \varrho_j + \sum_{i<j<k} \varrho_i \varrho_j \varrho_k - \sum_{i<j<k<l} \varrho_i \varrho_j \varrho_k \varrho_l. \tag{4.2.7}$$

It is easy to show that $\varrho(\mathbf{r}) = 0$ if $\mathbf{r}$ is within the VWS of the molecule, $\varrho(\mathbf{r}) = 1$ if $\mathbf{r}$ is in the exterior of the SAS, $0 < \varrho(\mathbf{r}) < 1$ if $\mathbf{r}$ is between the VWS and the SAS. We define the complementary function $\bar{\varrho} = 1 - \varrho$. Then (4.2.4) can be rewritten as

$$R_i^{-1} = \frac{1}{4\pi} \int_{\mathbb{R}^3} \frac{\bar{\varrho}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r}. \tag{4.2.8}$$

Differentiating both sides of (4.2.8), one gets

$$\begin{aligned} -\frac{1}{R_i^2} \frac{\partial R_i}{\partial \mathbf{x}_\alpha} &= \frac{1}{4\pi} \int_{\mathbb{R}^3} \frac{\partial}{\partial \mathbf{x}_\alpha} \left(\frac{\bar{\varrho}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4} \right) \, \mathrm{d}\mathbf{r} \\ &= \frac{1}{4\pi} \left(\int_{\mathbb{R}^3} \frac{\frac{\partial}{\partial \mathbf{x}_\alpha} \bar{\varrho}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r} + \int_{\mathbb{R}^3} \bar{\varrho}(\mathbf{r}) \frac{\partial}{\partial \mathbf{x}_\alpha} \left(\frac{1}{|\mathbf{r} - \mathbf{x}_i|^4} \right) \, \mathrm{d}\mathbf{r} \right). \end{aligned} \tag{4.2.9}$$

So

$$\frac{\partial R_i}{\partial \mathbf{x}_\alpha} = -\frac{R_i^2}{4\pi} \left(\int_{\mathbb{R}^3} \frac{\frac{\partial}{\partial \mathbf{x}_\alpha} \bar{\varrho}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r} + \int_{\mathrm{ex}} \frac{\partial}{\partial \mathbf{x}_\alpha} \frac{1}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r} \right). \tag{4.2.10}$$

For the first integral in (4.2.10),

$$\frac{\partial}{\partial \mathbf{x}_\alpha}\bar{\varrho} = -\frac{\partial}{\partial \mathbf{x}_\alpha}\varrho = -\frac{\partial \varrho_\alpha}{\partial \mathbf{x}_\alpha}(1 - \sum_j \varrho_j + \sum_{j<k} \varrho_j \varrho_k - \sum_{j<k<l} \varrho_j \varrho_k \varrho_l) = -\frac{\partial \varrho_\alpha}{\partial \mathbf{x}_\alpha} g_\alpha,$$

where j, k, l are the atoms overlapping with atom α, $g_\alpha = 1 - \sum_j \varrho_j + \sum_{j<k} \varrho_j \varrho_k - \sum_{j<k<l} \varrho_j \varrho_k \varrho_l$, and

$$\frac{\partial \varrho_i}{\partial \mathbf{x}_\alpha}(\mathbf{r}) = \begin{cases} 0, & x \le a_\alpha \\ (\frac{6}{w^3}(x-a_\alpha)^2 - \frac{6}{w^2}(x-a_\alpha))\frac{\mathbf{x}_\alpha - \mathbf{r}}{x}, & a_\alpha < x < a_\alpha + w \\ 0, & x \ge a_\alpha + w \end{cases}$$

with $x = ||\mathbf{r} - \mathbf{x}_\alpha||$. Noticing that $\frac{\partial \varrho_\alpha}{\partial \mathbf{x}_\alpha} \neq 0$ only if $a_\alpha < |\mathbf{r} - \mathbf{x}_\alpha| < a_\alpha + w$, thus the first integral in (4.2.10) is simplified as

$$\int_{|\mathbf{r}-\mathbf{x}_\alpha| = a_\alpha}^{|\mathbf{r}-\mathbf{x}_\alpha| = a_\alpha + w} -\frac{\partial \varrho_\alpha}{\partial \mathbf{x}_\alpha} g_\alpha(\mathbf{r}) \frac{1}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r}. \tag{4.2.11}$$

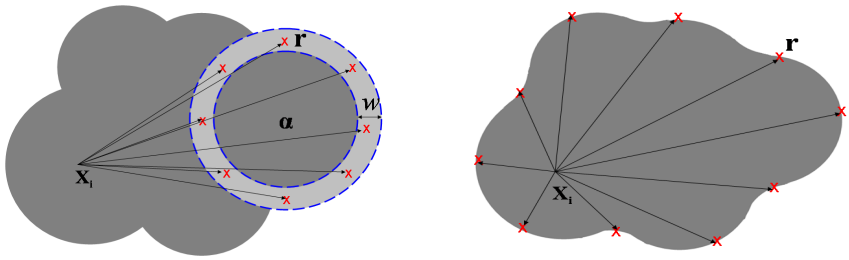
The integration domain of (4.2.11) is a regular spherical shell of width w around atom α (Figure 4.1(a)). We transform to the spherical coordinate system:

$$\begin{cases} x = x_\alpha + (a_\alpha + r)\cos\theta \sin\phi \\ y = y_\alpha + (a_\alpha + r)\sin\theta \sin\phi \\ z = z_\alpha + (a_\alpha + r)\cos\phi \end{cases}$$

where $(r, \theta, \phi) \in [0, w] \times [0, 2\pi] \times [0, \pi]$. We sample r, θ, ϕ by using the 2-point Gaussian quadrature nodes in each dimension. For all the atoms in the molecule, they share the same set of sampling points (r, θ, ϕ).

The second integral in (4.2.10) is nonzero only if $i \equiv \alpha$. In that case

$$\int_{\mathrm{ex}} \frac{\partial}{\partial \mathbf{x}_i} \frac{1}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r} = -\int_{\mathrm{ex}} \frac{\partial}{\partial \mathbf{r}} \frac{1}{|\mathbf{r} - \mathbf{x}_i|^4} \, \mathrm{d}\mathbf{r}. \tag{4.2.12}$$



(a) shell (b) surface

Figure 4.1: When computing the derivatives of the Born radii $\frac{\partial R_i}{\partial \mathbf{x}_\alpha}$, the quadrature points of the first integral are points within a spherical shell around atom α, as shown in (a), whereas the second integral is necessary when $i \equiv \alpha$ and the quadrature points are points on the surface, as shown in (b). The dark region represents the molecule, the light grey region is the shell of width w around atom α.

We compute each component of (4.2.12) individually and convert them to the surface integrals as we show in Figure 4.1(b) by using the divergence theorem:

$$-\int_{\text{ex}} \frac{\partial}{\partial x}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r} = \int_\Gamma \frac{n_x(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}S = \sum_{k=1}^{N} \frac{w_k n_x^k}{|\mathbf{r}_k-\mathbf{x}_i|^4} \tag{4.2.13}$$

$$-\int_{\text{ex}} \frac{\partial}{\partial y}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r} = \int_\Gamma \frac{n_y(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}S = \sum_{k=1}^{N} \frac{w_k n_y^k}{|\mathbf{r}_k-\mathbf{x}_i|^4} \tag{4.2.14}$$

$$-\int_{\text{ex}} \frac{\partial}{\partial z}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}\mathbf{r} = \int_\Gamma \frac{n_z(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}S = \sum_{k=1}^{N} \frac{w_k n_z^k}{|\mathbf{r}_k-\mathbf{x}_i|^4} \tag{4.2.15}$$

where the quadrature weights and points $(w_k, \mathbf{r}_k)$ and the unit normals (n_x^k, n_y^k, n_z^k) are the same as those used in Section 3.4. We compute (4.2.13), (4.2.14), and (4.2.15) by applying the fast summation method with the coefficients $c_k = w_k n_x^k,\ w_k n_y^k,\ w_k n_z^k$, respectively.

To compute the force acting on each of the M atoms, we need to compute (4.2.13), (4.2.14), and (4.2.15) for $i = 1, \ldots, M$. By using the fast sum-

mation algorithm, the computational complexity of this part is $O(N + M + n^3 \log n)$, which is the same as the complexity of the energy computation. To compute (4.2.11), since the shell is narrow, only a small number of atoms have non-zero densities in this region, therefore the complexity of computing (4.2.11) for a fixed α and for $i = 1, \ldots, M$ is $O(M)$. Moreover, since the integrand in (4.2.11) is very small if atom i and atom α are far apart, we use a cut-off distance d_0 in our computation and compute (4.2.11) only when the distance between i and α is less than d_0. Therefore, the overall time complexity of computing (4.2.10) is $O(N + M + n^3 \log n)$.

4.3 Results

For the test proteins 1ANA, 1PPE_l, and 1CGI_l, we compute the solvation force $\mathbf{F}_\alpha^{\text{elec}}$, for $\alpha = 1, \ldots, M$, where $M = 249$, 436, and 852, respectively. We show the computation time of each protein in Table 4.1.

Protein ID	M	N	t_1 (s)	t_2 (s)	T_{total} (s)
1ANA	249	6,676	66.05	0.14	66.19
1PPE_l	436	5,548	59.55	0.56	60.11
1CGI_l	852	6,792	68.71	3.27	71.98

Table 4.1: The computation time of the force calculation: M is the number of atoms, N is the number of triangles of the surface triangular mesh, t_1 is the time of computing (4.2.12) for $i = 1, \ldots, M$ and t_2 is the time of computing the other terms in (4.2.2) except $\frac{\partial R_i}{\partial \mathbf{x}_i}, i = 1, \ldots, M$. T_{total} is the total computation time.

For the test proteins, we compute the solvation force on each atom and

score the atoms based on the magnitude of the forces. We select the top 5% of the atoms which have high scores and color them in red. We also select the bottom 5% of the atoms with low scores and color them in blue. In Figure 4.2, we show the comparison of the active/stable atoms detected by the our GB method and by Amber. For 1CGI, 83.3% of the active atoms (red color) found by Amber are also detected by the our S-GB method and 81% of the stable atoms (blue color) tagged by Amber are also tagged in our S-GB method. For 1HAI, the similarities are 45.7% for the active atoms and 57.1% for the stable atoms. For 1PPE, the similarities are 72.7% for the active atoms and 68.2% for the stable atoms.

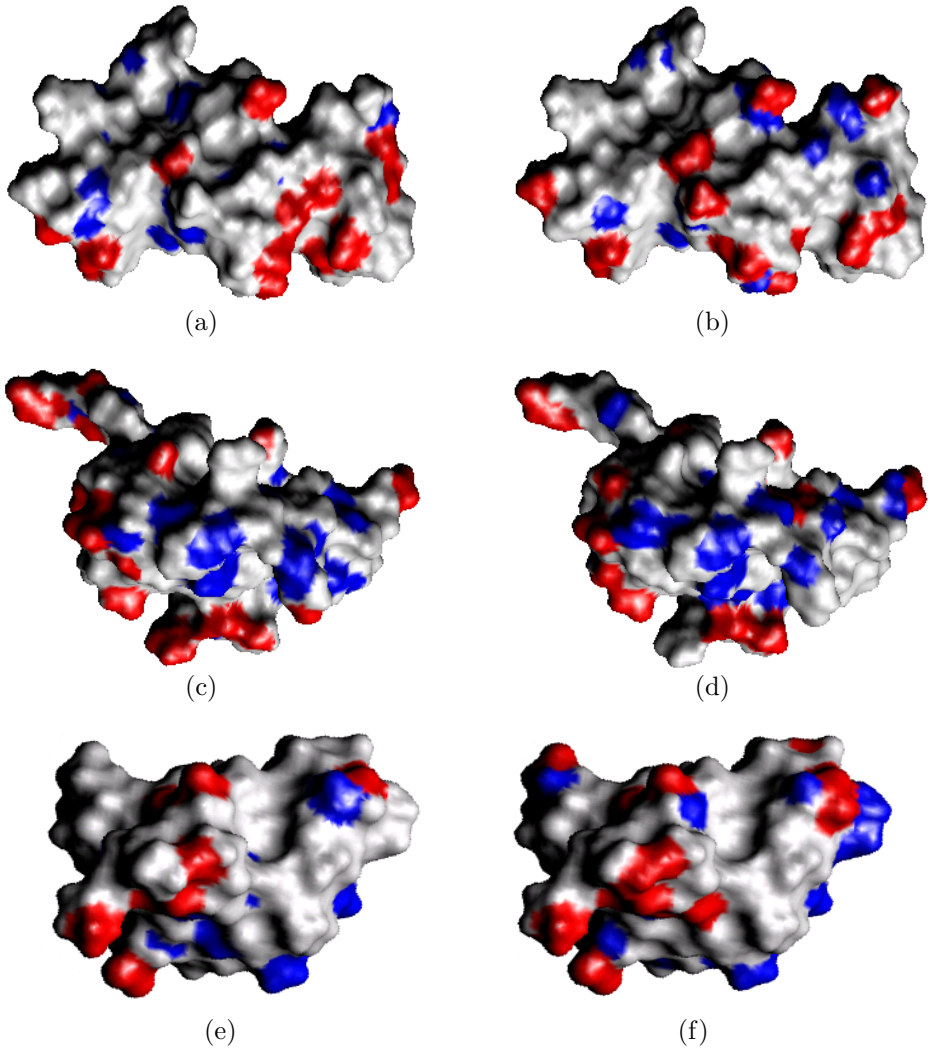


Figure 4.2: Atoms that have the strongest electrostatic solvation force (top 5%) are colored in red. Atoms that have the weakest electrostatic solvation force (bottom 5%) are colored in blue. (a), (c), (e) are generated from our GB method and (b), (d), (f) are generated from Amber.

Chapter 5

Coarse Grained Free Energy of Proteins and Macromolecules

5.1 Introduction

In biological systems, many phenomena such as protein folding occur on long time scales from microseconds to milliseconds and may involve length scales as large as micrometers [61]. Despite the fact that the traditional all-atom (AA) MD simulation reveals the details of the folding process, the AA MD simulation is currently limited to a few tens of nanoseconds and a few hundreds of angstroms due to its low computational efficiency [62]. To increase the time and length scales of the MD simulations, one feasible way is reducing the level of detail of the system. The coarse grained (CG) models which represent a group of atoms with similar physical properties by a single bead can significantly reduce the size of the system, and hence has become an increasingly popular approach to reproduce large-scale protein motions. Recently the CG models have been applied to the simulation of the large-scale motions of individual proteins [63], ribosomes [64], virus capsids [65, 66], lipid bilayers [67] and lipoproteins [68, 69].

In this chapter, we discuss the state of the art in CG modeling in Section

5.2, investigate a hierarchical error-bounded CG model generation in Section 5.3. In Section 5.4 we show some examples of the CG models of macromolecules at different resolutions and show that the electrostatic solvation energy results of the CG models match those of the AA models.

5.2 Prior work

5.2.1 Coarse grained modeling

Coarse grained models are basically classified into two families: lattice models [70–72] and non-lattice models [68, 73]. Typically in a lattice model, a set of CG beads are randomly arranged on a three-dimensional lattice grid and then the Monte Carlo (MC) simulation is conducted with the lattice model, aiming at finding the configuration with the lowest total energy. Although the lattice MC computation is efficient, the lattice models are less realistic and versatile.

The prototype of the non-lattice CG model was developed by Smit and coworkers in [74]. In their model a lipid molecule is partitioned into a hydrophilic 'head' and a hydrophobic 'tail' and all the particles interact via a sort of simplified Lennard-Jones potential, that is, it is either repulsive for oil-water interactions or attractive for particles of the same type. A systematic CG approach was developed by Shelley et al [75, 76] in order to quantitatively reproduce the behavior of phospholipid. In their approach, the CG model is systematically parameterized to mimic the existing force fields or statistical mechanical properties obtained from the atomistic models. The MD simula-

tion based on their model semiquantitatively agrees with the atomistic results. Marrink et al [73] improve the CG model of lipid by classifying the CG beads into different types according to properties such as hydrophobicity, hydrogen bonding capacity, and charge. They also simplify the force field by using a short-range cutoff (1.2 nm) for the non-bonded interactions and fewer parameters.

For polypeptide chains (i.e. proteins), several levels of CG representations have been developed. The Gō model is one of the earliest and simplest models where the polypeptide chain is represented by a chain of C_α atoms with attractive or repulsive non-bonded interactions only [77]. This model has been developed in many aspects [78–80] and was recently used in modelling the flapping opening dynamics of HIV-1 protease [81]. The side chains (SC) are included in the two-bead models, where each SC bead is assumed to be at the center of mass of the heavy atoms in the side chain [82, 83]. The two-bead models have been applied to the simulation of peptide binding to HIV-1 protease [84], lipoprotein assembling [68], and so on. In the C_α-SC-Pep model [85], an additional interaction center (Pep) is added on the backbone in the middle of the C-N peptide bond which strongly improves the orientation-dependent potentials. In [86] extended side chains (such as Arg, Lys, etc.) are represented by two beads in order to have CG beads of about the same size. A similar model is applied to the study of protein-protein docking [87]. A four-bead model is given in [88] which explicitly represent the three heavy atoms of the backbone and the hydrogen bonds. Most recently a multiresolution CG

model is developed in [89] which allows coarser and finer representations in different parts of the macromolecule and therefore fixes the deficiency of assigning each CG bead to the same number of atoms. In [90], an ellipsoid CG model is presented.

5.2.2 Coarse grained force field

After the molecule is partitioned into a reduced system, the next important issue is to build an effective force field to describe the interactions between CG beads. The molecular mechanics of a protein is an interplay of different interactions such as covalent bonds, dihedral angles, hydrogen bonds, electrostatic interactions, van der Waals interactions, hydrophobic interactions, etc. The fewer degrees of freedom at the CG level make it difficult to accurately describe the properties and the energies of a protein molecule. In general, one needs to first define an analytical form for the interaction energy and then parameterize the form such that certain properties coincide with those seen in the atomic level simulation. These properties include the knowledge-based statistical potential such as the potential of mean force [75], the effective thermodynamic properties [75], the structural properties such as the area [73] and the radial distribution function [91], and the trajectory and force data from the MD simulation [92]. Methods for finding the parameters in the CG effective potentials involve the iterative Boltzmann inversion method which is one of the mapping techniques to derive mainly the non-bonded potentials [93], the inverse Monte Carlo method which one can use to systematically derive

all effective interaction potentials [94], and the least squares fitting which allows one to build a realistic force field from *ab initio* MD simulations at low computational cost [92].

5.3 CG model generation

5.3.1 Clustering

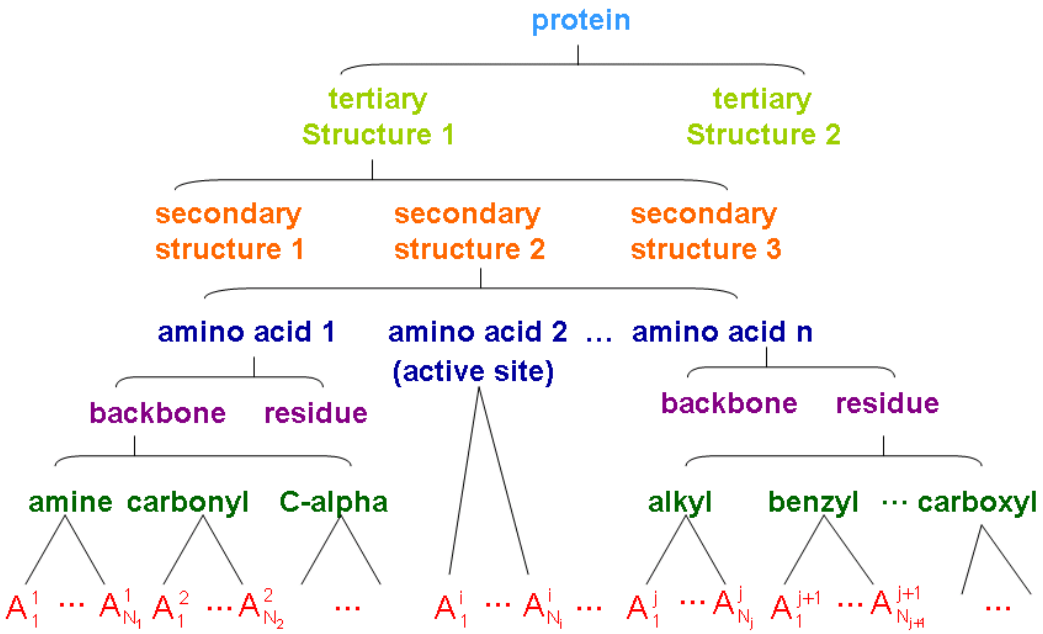


Figure 5.1: Hierarchical representation of a large protein structure.

In Figure 5.1, we show a hierarchical representation of a large protein structure. From bottom to top:

1. **Atomic structure.** At the bottommost level, each atom, such as carbon, oxygen, nitrogen, hydrogen, etc. can be discriminated in space with resolution less than 2.5 Å.

2. **Functional group.** Specific groups of atoms that characterize the chemical reaction of a molecule form a functional group, e.g. the acidic carboxyl (-COOH), the hydrophobic benzyl (-C_6H_5), etc.

3. **Residue and backbone.** Twenty standard amino acids are used by cells to build proteins. Each amino acid has a backbone (containing an amine, an alpha carbon, and a carboxyl functional group) and a side chain (also known as residue) attached to the alpha carbon (Figure 5.2).

4. **Secondary structure.** A group of amino acids construct the secondary structure, such as α-helices, β-sheets and turns. The resolution of this level is about 5 Å to 10 Å.

5. **Tertiary structure.** Group of secondary structure constitute the tertiary structure, also called motifs. Motifs may involve several chains. The resolution of this level is about 10 Å to 25 Å.

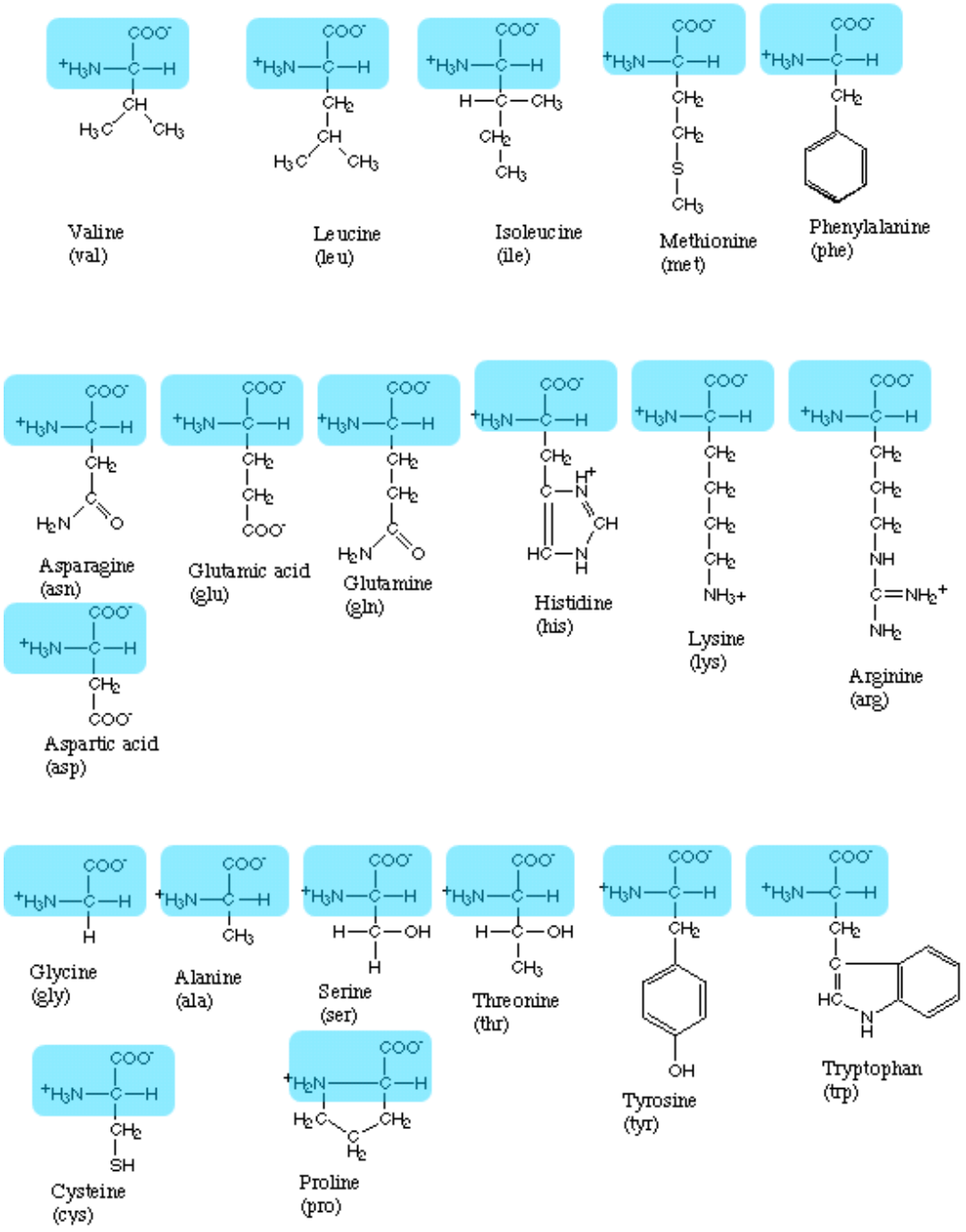
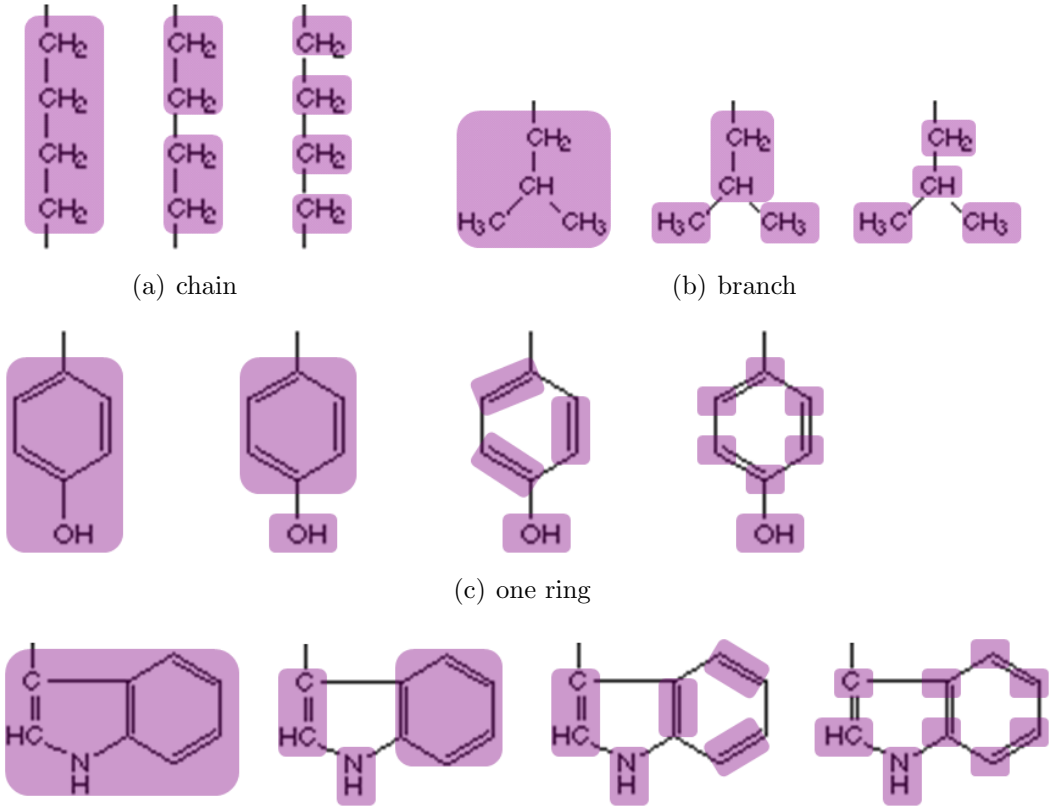


Figure 5.2: Twenty standard amino acids. The blue rectangle represents the backbone and the remaining is the side chain.

According to the tiered structure, one can cluster the atoms into CG beads at different levels. Even though at the highest level, the structure is less complex, too much information of the protein gets lost. Hence, in our current research, as we attempt to simplify the model while still keeping some detail of the molecules, we build the CG model up to the amino acid level.

At the amino acid level, the whole amino acid is represented by one bead. At the backbone-residue level, except for glycine which has only one hydrogen on the residue, the atoms belonging to the backbone and the residue are grouped as a bead. At the next level, the backbone is divided into an amine group, a carbon-α attached with a hydrogen, and an acid group. The side chain can also be divided into functional groups such as alkyl, acid, alcohol, amine, carboxamide, benzyl, phenol, imidazole, indole, etc. Each functional group is represented by one CG bead. At the last level, every atom is explicitly represented.

Notice that using a single bead to represent the large functional groups may induce a large error in geometry, while the AA model costs too much computational time. Therefore we introduce intermediate CG models for these large functional groups.



(a) chain

(b) branch

(c) one ring

(d) two rings

Figure 5.3: (a) Chain structure of butyl which has four alkyls. (b) Branch structure of isobutyl which has two carbon alkyls on the main branch and one alkyl on each of the two minor branches. (c) Ring structure of phenol. (d) Double ring structure of indole. For each structure (a-d), from left to right, we show how the structure is hierarchically grouped into CG beads. The purple rectangles represent the CG beads.

For the functional groups containing a long chain, e.g. the butyl on the side chain of lysine which has four alkyls on the chain (Figure 5.3(a)), we iteratively cut the chain into halves until there is only one heavy atom in each bead. For the branched functional groups, e.g. the isobutyl group on the

side chain of leucine (Figure 5.3(b)), we cluster the atoms on the main branch (the branch that is attached to the backbone) into one bead and the atoms on each minor branch into one bead. Next we apply the clustering scheme of the chain structure to each branch until we have one heavy atom per bead. For the functional groups containing a ring structure, e.g. the phenol in tyrosine (Figure 5.3(c)), we first identify the atoms on the ring and cluster them into one bead. At the next level, we cut off the ring at the single bonds and group the atoms connected by a double bond, along with the associated hydrogens, into one bead. Lastly we cut the double bonds so that each of the heavy atoms on the ring is represented by a CG bead. The functional group indole in tryptophan has a complicated structure of two rings (Figure 5.3(d)). For this case we cut the first ring (we count starting from the backbone end) at its single bond followed by cutting the single bond of the second ring.

The hierarchical clustering we presented is based on knowledge of the fundamental conformations, chemical properties, and geometry features of the proteins. From this clustering, one can design a CG model at different resolutions. The level of resolution depends on the requirements for the specific problem.

5.3.2 Center and radius

Given the clustering of the atoms of a molecule, we compute the new centers and radii $\{(\mathbf{x}'_i, a'_i)\}_{i=1}^{M'}$ for the CG beads such that the new molecular surface of the CG model is close to the that of the AA model. Instead of

optimizing the locations and sizes of all of the beads globally, we optimize each bead locally and individually. Our scheme is to compute the center and the radius of each CG bead such that locally the molecular surface of the CG bead and that of the atoms belonging to the bead agrees as much as possible.

Consider a group of atoms $A_1, \ldots, A_m$, which are clustered into one CG bead. The centers and the radii of the atoms and the bead are $\{(\mathbf{x}_i, a_i)\}_{i=1}^m$ and $(\mathbf{x}', a')$. Let $MS_0(\mathbf{x}_1, \ldots, \mathbf{x}_m, a_1, \ldots, a_m)$ denote the molecular surface of the atomistic structure and $MS_c(\mathbf{x}', a')$ denote the molecular surface of the CG bead. We model MS_0 by using the method introduced in [36], where it is modeled as the zero level set of the function:

$$g(\mathbf{x}) = \sum_{i=1}^{m} e^{-\beta_i(||\mathbf{x}-\mathbf{x}_i||^2 - a_i^2)} - 1. \tag{5.3.1}$$

where β_i is the decay rate of the Gaussian function. As a preliminary value for β_i, we use 2.3 Å^{-2} for all the atoms to ensure that the zero level set of this function is close to the solvent excluded surface (SES). We construct MS_c in the same way except that function (5.3.1) is replaced with a single Gaussian function

$$f(\mathbf{x}) = e^{-\beta(||\mathbf{x}-\mathbf{x}'||^2 - a'^2)} - 1. \tag{5.3.2}$$

In order to minimize $||MS_c - MS_0||$, i.e. to force the zero sets of function (5.3.1) and (5.3.2) to be as close as possible, we look for $\mathbf{x}'$ and a' such that the function

$$\chi = \frac{1}{2} \sum_{j=1}^{n} [f(\mathbf{r}_j) - g(\mathbf{r}_j)]^2 \tag{5.3.3}$$

is minimized, where $\mathbf{r}_j \in MS_0(\mathbf{x}_1, \ldots, \mathbf{x}_m, a_1, \ldots, a_m)$. Because $g(\mathbf{r}_j) = 0$, (5.3.3) is simplified to

$$\chi = \frac{1}{2}\sum_{j=1}^{n} f^2(\mathbf{r}_j) = \frac{1}{2}\sum_{j=1}^{n}\left[e^{-\beta(||\mathbf{r}_j - \mathbf{x}||^2 - a^2)} - 1\right]^2. \tag{5.3.4}$$

Before we solve the minimization problem (5.3.4), we introduce some notations. Let $\mathbf{x}' = (x', y', z')$. Define a new vector $\mathbf{v} = (x', y', z', a')$ and v_i denotes the ith component of $\mathbf{v}$. Let $f_j(\mathbf{v}) := f(\mathbf{r}_j)$ and $\mathbf{f}(\mathbf{v}) := (f_1(\mathbf{v}), \ldots, f_n(\mathbf{v}))$. Then we define a matrix

$$J(\mathbf{v}) = \begin{pmatrix} \frac{\partial f_1}{\partial v_1} & \frac{\partial f_1}{\partial v_2} & \frac{\partial f_1}{\partial v_3} & \frac{\partial f_1}{\partial v_4} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial v_1} & \frac{\partial f_n}{\partial v_2} & \frac{\partial f_n}{\partial v_3} & \frac{\partial f_n}{\partial v_4} \end{pmatrix}.$$

Then the gradient of χ is

$$\nabla\chi(\mathbf{v}) = \sum_{j=1}^{n} f_j(\mathbf{v})\nabla f_j(\mathbf{v}) = J^T(\mathbf{v})\mathbf{f}(\mathbf{v}), \tag{5.3.5}$$

and the Hessian matrix of χ is

$$\begin{aligned} H(\mathbf{v}) &= \sum_{j=1}^{n}(\nabla f_j(\mathbf{v})\nabla f_j^T(\mathbf{v}) + f_j(\mathbf{v})\nabla^2 f_j(\mathbf{v})) \\ &= J^T(\mathbf{v})J(\mathbf{v}) + S(\mathbf{v}), \end{aligned} \tag{5.3.6}$$

where $S(\mathbf{v}) = \sum_{j=1}^{n} f_j(\mathbf{v})\nabla^2 f_j(\mathbf{v})$, $\nabla^2 f_j(\mathbf{v})$ is a 4×4 matrix and its ik-th entry is $\frac{\partial^2 f_j}{\partial v_i \partial v_k}$. For a successful model, $f_j(\mathbf{v})\nabla^2 f_j(\mathbf{v})$ tends to cancel out when summed over j, so sometimes $S(\mathbf{v})$ is ignored in practice [95].

We solve the nonlinear least squares problem (5.3.4) by using the standard Levenberg-Marquardt algorithm (LMA) [96] which combines the steepest

descent (SD) method and Newton's method. Let $\mathbf{d}^k$ denote $-\nabla\chi(\mathbf{v}^k)$ and H^k denoted $H(\mathbf{v}^k)$.

In the SD method, at a point $\mathbf{v}^k$, the next point $\mathbf{v}^{k+1}$ is searched for in the direction opposite to the gradient direction $\nabla\chi(\mathbf{v}^k)$:

$$\mathbf{v}^{k+1} = \mathbf{v}^k + \alpha^k \mathbf{d}^k, \qquad (\alpha^k > 0). \tag{5.3.7}$$

The step length α^k is chosen such that χ reaches a minimum at $\mathbf{v}^{k+1}$ along the direction $\mathbf{d}^k$, i.e. the directional derivative of χ at $\mathbf{v}_{k+1}$ is zero:

$$\frac{d\chi}{d\alpha^k}(\mathbf{v}^{k+1}) = \nabla\chi(\mathbf{v}^{k+1}) \cdot \frac{d}{d\alpha^k}\mathbf{v}^{k+1} = \nabla\chi(\mathbf{v}^{k+1}) \cdot \mathbf{d}^k = 0. \tag{5.3.8}$$

From (5.3.8) we see that the following search direction $\mathbf{d}^{k+1}$ is orthogonal to $\mathbf{d}^k$, therefore points in the SD method move along a zig-zag path towards the minimum, and consequently the convergence is very slow.

In Newton's optimization method, around $\mathbf{x}^k$, (5.3.4) is approximated by a truncated Taylor series:

$$\chi(\mathbf{v}) \approx \chi(\mathbf{v}^k) + \nabla\chi(\mathbf{v}^k) \cdot (\mathbf{v} - \mathbf{v}^k) + \frac{1}{2}(\mathbf{v} - \mathbf{v}^k)^T H^k (\mathbf{v} - \mathbf{v}^k). \tag{5.3.9}$$

Differentiating (5.3.9) we get the gradient of χ:

$$\nabla\chi(\mathbf{v}) \approx \nabla\chi(\mathbf{v}^k) + \frac{1}{2}H^k (\mathbf{v} - \mathbf{v}^k) + \frac{1}{2}(H^k)^T (\mathbf{v} - \mathbf{v}^k) = \nabla\chi(\mathbf{v}^k) + H^k (\mathbf{v} - \mathbf{v}^k), \tag{5.3.10}$$

in which H^k is symmetric. $\chi(\mathbf{v})$ reaches its local minimum at $\mathbf{v}^*$ if $\nabla\chi(\mathbf{v}^*) = 0$ and the Hessian matrix $H(\mathbf{v}^*)$ is positive definite. By solving (5.3.10) for $\nabla\chi(\mathbf{v}) = 0$, we get the iterative formula

$$\mathbf{v}^{k+1} = \mathbf{v}^k + [H^k]^{-1}\mathbf{d}^k. \tag{5.3.11}$$

Even though the convergence of Newton's method could be quadratic, the performance largely depends on the quality of the Hessian. For example, in order to obtain the next point $\mathbf{v}^{k+1}$ in the descent direction, we need to require $(\mathbf{v}^{k+1} - \mathbf{v}^k) \cdot \nabla\chi(\mathbf{v}^k) < 0$, which is equivalent to

$$-(\mathbf{v}^{k+1} - \mathbf{v}^k) \cdot H^k \, (\mathbf{v}^{k+1} - \mathbf{v}^k) < 0. \tag{5.3.12}$$

If H^k is positive definite, then the condition (5.3.12) is satisfied for all $\mathbf{v}^{k+1} \neq \mathbf{v}^k$. However when $\mathbf{v}^k$ is far from the minimum, H^k could be non positive definite, then one cannot guarantee that $\mathbf{v}^{k+1}$ is in the decreasing direction. Therefore usually it is desirable to use Newton's method only when the point is within a small neighborhood of the minimum.

The LMA interpolates the SD method and Newton's method so that it can quickly find a solution no matter where it starts. The iterative formula of the LMA is

$$\mathbf{v}^{k+1} = \mathbf{v}^k + [H^k + \mu \text{diag}(H^k)]^{-1}\mathbf{d}^k. \tag{5.3.13}$$

By adjusting μ, one can easily switch between the SD and Newton's method. When μ is very large, the matrix $H^k + \mu\text{diag}(H^k)$ is diagonally dominant, then (5.3.13) is nearly identical to the SD method (5.3.7), on the other hand, as $\mu \downarrow 0$, (5.3.13) continuously switches to Newton's method (5.3.11). Given a point $\mathbf{v}^k$ and an initial μ, one can solve $[H^k + \mu\text{diag}(H^k)]\delta\mathbf{v} = \mathbf{d}^k$ for $\delta\mathbf{v}$. Compute $\chi(\mathbf{v}^k + \delta\mathbf{v})$. If $\chi(\mathbf{v}^k + \delta\mathbf{v}) < \chi(\mathbf{v}^k)$, let $\mathbf{v}^{k+1} = \mathbf{v}^k + \delta\mathbf{v}$. Since $\mathbf{v}_{k+1}$ is approaching the minimum, one can try a smaller μ in the next iteration for a rapid convergence. On the other hand, if $\chi(\mathbf{v}^k + \delta\mathbf{v}) > \chi(\mathbf{v}^k)$, the search

direction is wrong. In this case one cannot use Newton's method and has to increase μ to use the SD method to force the search direction to be descent. The reason of using diag(H^k) in (5.3.13) rather than a unit matrix is that it allows a larger movement in the directions where the gradient is small. The steps of the LMA is given in Algorithm 1.

Algorithm 1 Levenberg-Marquardt Algorithm

given an initial guess $\mathbf{v}^0$, $\mu = 0.001$
compute $\chi(\mathbf{v}^0)$, $\mathbf{d}^0$, and H^0
$k \leftarrow 0$
loop
  solve $(H^k + \mu I)\delta\mathbf{v}^k = \mathbf{d}^k$ for $\delta\mathbf{v}^k$
  if $\chi(\mathbf{v}^k + \delta\mathbf{v}^k) > \chi(\mathbf{v}^k)$ **then**
    $\mu \leftarrow 10 * \mu$
  else
    $\mathbf{v}^{k+1} \leftarrow \mathbf{v}^k + \delta\mathbf{v}^k$
    if $\chi(\mathbf{v}^{k+1}) - \chi(\mathbf{v}^k) < \epsilon$ **then**
      exit loop
    else
      compute $\mathbf{d}^{k+1}$ and H^{k+1}
      $k \leftarrow k + 1$
      $\mu \leftarrow 0.1 * \mu$
    end if
  end if
end loop
output $\mathbf{v}^{k+1}$

Besides LMA, there are other advanced algorithms for nonlinear least squares, such as the hybrid method which combines Newton's method and quasi-Newton's method and the "full Newton-type" method which keeps the second derivative term $S(\mathbf{v})$ in the Hessian matrix. However these methods are more complex than LMA. Hence we use LMA as the solver in our work.

5.3.3 Charge

One way of assigning the charges to the CG beads is the so-called "natural charges" method in which the charge of a CG bead is equal to the total charge of the associated atoms [87]. For the beads belonging to the charged amino acids such as the acidic amino acids (ASP^- and GLU^-), the basic amino acids (ARG^+, HIS^+, and LYS^+), and the terminal residues (NH_3^+ and COO^-), they have a net charge corresponding to the amino acids. Despite the fact that the natural charges model preserves the total charge of a protein, this simple charge distribution does not account for the interactions between charges.

In this work, our goal is to correctly estimate the electrostatic solvation energy by using the coarse-grained model. Therefore when finding the CG charges $\{q_i'\}_{i=1}^{M'}$, we try to minimize the difference between the electrostatic solvation energy of the CG model and that of the atomic level model. For both CG model and the atomic model, we compute the electrostatic solvation energy by using the same energy function (3.2.6). Hence mathematically the problem becomes finding $\{q_i'\}_{i=1}^{M'}$ such that the function

$$\chi_2 := \left(\sum_{i=1}^{M'} \sum_{j=1}^{M'} \frac{q_i' q_j'}{f_{GB}'} - \sum_{i=1}^{M} \sum_{j=1}^{M} \frac{q_i q_j}{f_{\mathrm{GB}}} \right)^2 \tag{5.3.14}$$

is minimized subject to the constraint that $\sum_{i=1}^{M'} q_i' = \sum_{i=1}^{M} q_i$, i.e. the total charge of the molecule does not change. f_{GB} denotes

$$f_{\mathrm{GB}} = \left[r_{ij}^2 + R_i R_j \exp(-\frac{r_{ij}^2}{4R_i R_j}) \right]^{\frac{1}{2}},$$

and f'_{GB} denotes

$$f'_{\mathrm{GB}} = \left[(r'_{ij})^2 + R'_i R'_j \exp(-\frac{(r'_{ij})^2}{4R'_i R'_j})\right]^{\frac{1}{2}},$$

where r'_{ij} are the distances between the CG beads, R'_i are the effective Born radii of CG beads which can be computed in the same way as R_i in (3.3.10). We solve the nonlinear optimization problem (5.3.14) by using the LM algorithm introduced in Section 5.3.2.

5.4 Examples and Results

For the twenty amino acids, we create a table for each of them which provides the clustering information of the amino acid at all levels. We have developed a program to compute the optimized positions, radii, and charges of the quasi atoms of a protein, at any level of clustering the user defines. In Figure 5.4 we show the molecular surface of the lysine at four levels of coarsening, where from left to right the atoms are grouped into 2, 5, 6, 8 beads, and the atomic model, respectively. The surfaces of the CG model are generated as the Gaussian surface defined in Section 5.3.2 based on the optimized centers and radii of of the beads. The same comparison is done for tyrosine which has a phenol group on its side chain. In Figure 5.5, from left to right we show four level of CG models with 2, 6, 8, and 11 beads, respectively, and the atomic level surface.

We measure the error of the CG surfaces MS_{c} compared with the surface of the atomic model MS_0 as the one-way Hausdorff distance from MS_{c}

to MS_0:

$$\epsilon = \text{dist}(MS_c, MS_0) = \max_{\mathbf{x} \in MS_c} \min_{\mathbf{y} \in MS_0} ||\mathbf{x} - \mathbf{y}||.$$

In Table 5.1 We show the error of MS_c of lysine at different levels of coarsening. Similarly, in Table 5.2 we show the CG surface error of tyrosine. The error decreases as the clustering becomes finer and finer.

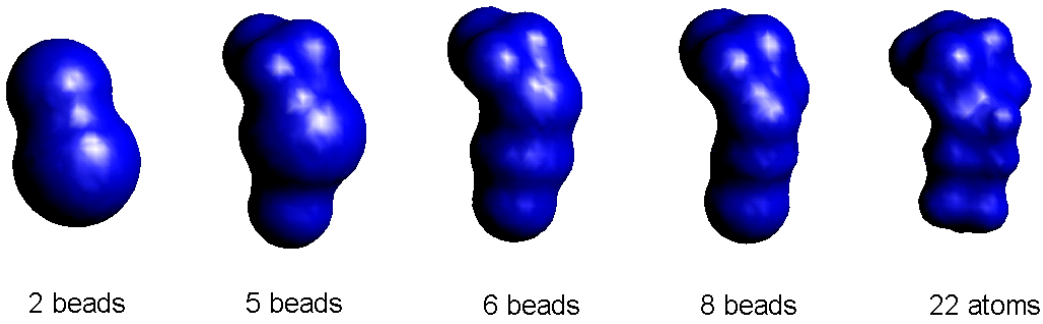


Figure 5.4: The molecular surface of lysine at different coarse grained levels. From left to right, the atoms in lysine are grouped into 2, 5, 6, 8 beads. The very right one is the surface at the atomic level.

No. of Beads	2	5	6	8
ϵ	1.420	1.148	0.608	0.606

Table 5.1: The error of four CG molecular surfaces of lysine at different level of clustering. The error ϵ is defined as the one-way Hausdorff distance from the CG molecular surface MS_c to the atomic molecular surface MS_0.

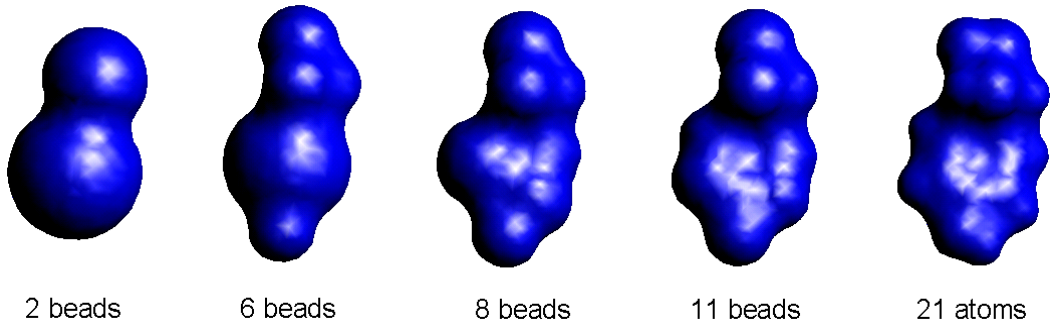


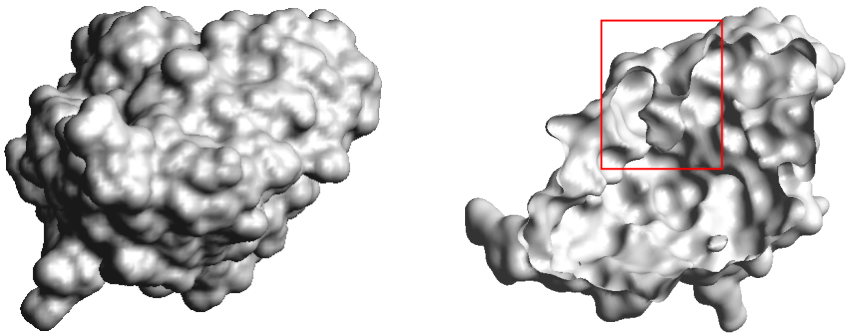
Figure 5.5: The molecular surface of tyrosine at different coarse grained level. From left to right, the atoms in lysine are grouped into 2, 6, 8, 11 beads. The very right one is the surface at the atomic level.

No. of Beads	2	6	8	11
ϵ	1.649	1.506	0.724	0.479

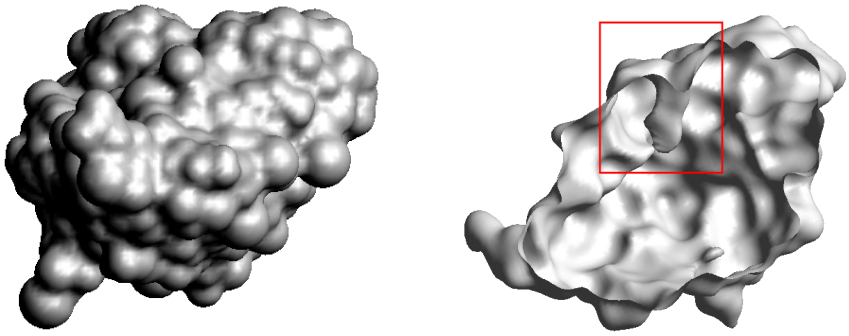
Table 5.2: The error of four CG molecular surfaces of tyrosine at different level of clustering. The error ϵ is defined as the one-way Hausdorff distance from the CG molecular surface MS_{c} to the atomic molecular surface MS_0.

In Figure 5.6, we show the coarse grained model of a real example, an acetylcholinesterase (AChE), at three different levels. At the AA level there are 8340 atoms. We group the atoms into 2534 beads where each backbone is represented by three beads and the atoms on each side chain is clustered into two beads except GLY where there is no bead on the side chain and ALA where there is only one bead on the side chain. The surface error of this model compared with the AA surface is 2.89 Å. We further simplify the CG model by grouping the atoms into 1035 beads where there is one bead per backbone and one bead per side chain except GLY. The surface error compared with

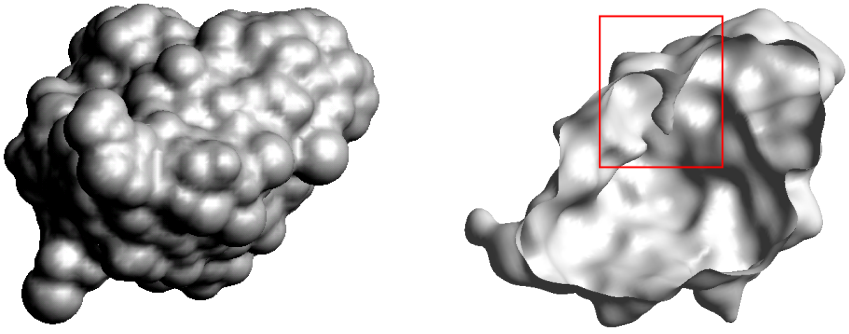
the AA surface is 3.57 Å. It is known that AChE has an important pocket structure underneath its surface (Figure 5.6(a)) and the active site of AChE locates exactly at the end of the pocket. For a CG model with correct topology, it must have the same structure. In Figure 5.6(c) and Figure 5.6(b), we see that both the 2-bead CG model and 5-bead CG model preserve the pocket structure. So our CG generation not only minimizes the distance between the AA surface and the CG surface, but also respects the topology of the molecules.



(a) 8340 atoms



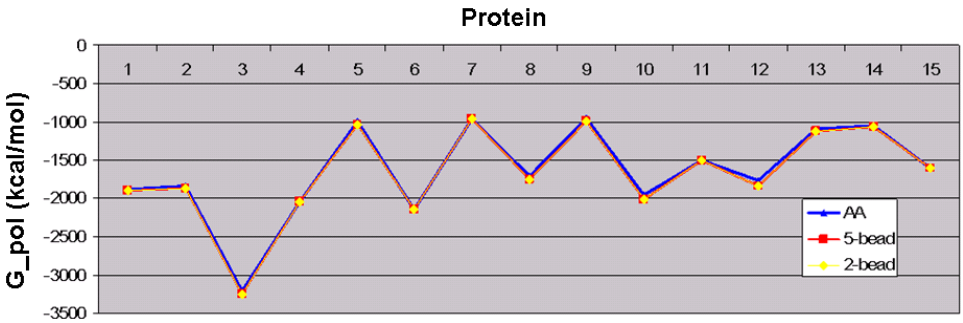
(b) 2534 beads



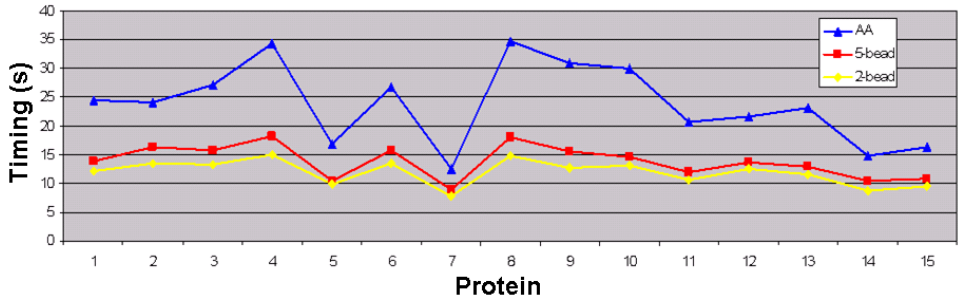
(c) 1035 beads

Figure 5.6: The coarse grained model of AChE at three different levels. (a) The molecular surface of the AA model (8340 atoms). (b) The molecular surface of the 5-bead CG model (2534 beads). (c) The molecular surface of the 2-bead CG model (1035 beads). The right column shows the pocket found in the AA model and the CG models.

We apply the CG model to the energy calculation. For a set of test proteins, we generate the 2-bead CG models where each amino acid is represented by a backbone bead and a side chain bead and the 5-bead CG models where each amino acid is represented by three backbone beads and two side chain beads (except GLY which has no beads on the side chain and ALA which has one bead on the side chain). We compare the electrostatic solvation energy G_{pol} of the AA model, the 2-bead CG model, and the 5-bead CG model for each protein, as shown in Figure 5.7(a). It turns out that the energy results of the CG models agree very well with the AA energy results. For the proteins we use in this test, the average relative error of G_{pol} computed by the 2-bead CG model compared with the AA model is 1.64926%, whereas the average relative error of G_{pol} computed by the 5-bead CG model compared with the AA model is 1.64923%. Since the complexity of the CG model is significantly reduced, the efficiency of the energy calculation is remarkably enhanced, as we show in Figure 5.7(b).



(a)

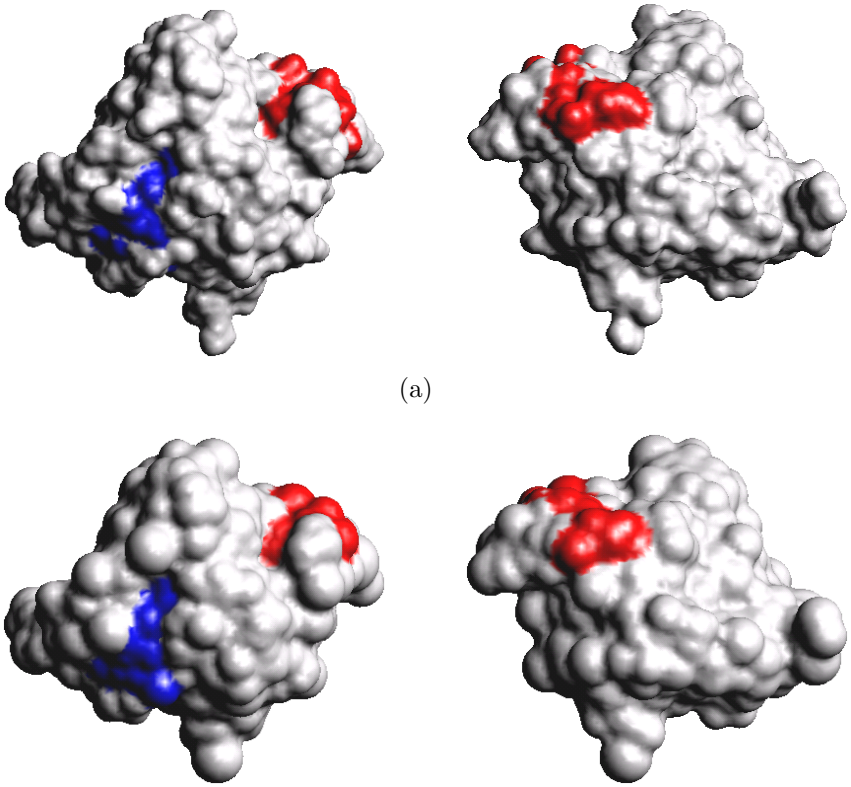


(b)

Figure 5.7: (a) The electrostatic solvation energy G_{pol} computed for the AA model, the 2-bead CG model, and the 5-bead CG model. (b) The time cost of computing G_{pol} for the AA model, the 2-bead CG model, and the 5-bead CG model.

Furthermore, we compute the electrostatic solvation force for the CG models and compare the force results with the AA model. For protein AChE, we compute the force for the AA model and color the atoms which have the strongest forces (top 5%) in red and the atoms which have the weakest forces (bottom 5%) in blue, as we show in Figure 5.8(a). We compute the force for the 2-bead CG model and color the beads by using the same criteria as the AA model, as we show in Figure 5.8(b). The forces computed from the CG

model match very well with the forces computed based on the AA model.



(a)

(b)

Figure 5.8: (a) The electrostatic solvation force computed for the AA model of AChE. The red color indicates the region which has the strongest solvation force (top 5%), whereas the blue color indicates the region which has the weakest electrostatic solvation force (bottom 5%). (b) The electrostatic solvation force computed for the 2-bead CG model of AChE. Beads are colored by using the same criteria as the AA model. The two columns represent two different views of the molecule.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

In this work we present a method to compute the electrostatic solvation energy of proteins based on the surface GB theory. The complexity of the surface GB computation depends on the number of sample points on the molecular surface and the number of atoms of the protein. We successively reduce the complexity the without losing the accuracy after making three major improvements.

First we present a method to generate an analytic model ASMS for the molecular surface. The ASMS model is smooth and close to the standard SES as long as the initial triangulation is based on the SES. In addition, it has dual implicit and parametric representations. The implicit representation enables us to flexibly generate different surfaces by varying the level sets. The parametric representation allows us to directly apply the ASMS to the numerical computations such as the numerical integrations involved in the finite element method or the boundary element method. Moreover and most importantly, the ASMS surface is generated based on a higher degree polynomial. Therefore, to obtain the same accuracy in the numerical integrations, the ASMS needs

fewer number of triangles (roughly one-third of the piecewise linear elements based on our test) than the piecewise linear models . Because of all of the advantages, the ASMS model is very suitable for use in the simulation of large biomolecules.

Second we introduce a fast summation based algorithm to calculate the effective Born radii and their derivatives in the generalized Born model of implicit solvation. The algorithm relies on a variation of the formulation for the Born radii and an additional analytical volumetric density function for the derivatives. For a system of M atoms and N sampling points on the molecular surface, the trivial way of computing the Born radii requires $O(MN)$ arithmetic operations, whereas with the aids of the Fourier expansion of the kernel functions of the Born radii (and their derivatives) and the NFFT algorithm which essentially approximates the complex exponentials in the NDFT by the DFT of a fast decaying smooth window function, the Born radii as well as their derivatives can be obtained at cost of $(M + N + n^3 \log n)$ where n is the number of frequencies in the Fourier expansion. We show that the error of the algorithm decreases as the mesh gets denser, or as any of the parameters σ, m, n increase.

Third we propose a coarse grained model to reduce the number of atoms in a protein. We provide a way of clustering the atoms at different levels of details based on the intrinsic hierarchical structures and the three dimensional shapes of the proteins. Given the information of the clustering, we represent each group of atoms as a CG bead. The positions and radii of the new CG

beads are carefully adjusted such that the molecular surface generated from the CG model is as close as possible to that generated from the atomic model. We also compute the charge of the CG beads by optimizing the GB functions such that the new CG model can best reproduce the electrostatics interactions of the atomic model. We show that for real molecules, the molecular surface of the CG model, even at a very coarse level, is still within a small error of the atomic surface, and the GB energy agrees very well with the atomic model.

We implement the above methods and apply to a large set of test proteins. The test set covers a significant range of protein structures of various sizes, amino acid compositions as well as folding topologies. The implementation results on the test set support our theory which makes us believe that the methods we presented in this dissertation are reliable and can be extended to more applications.

6.2 Future work

Some future studies that the author can foresee are listed in the following, among which some are currently ongoing:

- Extend the idea of the ASMS to quadrilateral meshes. The goal is to define a function in the parametric domain and guarantee that a level set of the function interpolates the mesh and is C^1 continuous. One possible way is to define the function as the Hermite interpolation of the functions defined on the faces of the prism where the latter are defined

as the Hermite interpolation of the functions defined on the edges of the prism.

- We think it will be valuable to apply our GB method to some of the interesting problems that demands efficient energy evaluations. One such problem is the protein-protein docking problem in which one protein probes along the surface of another protein complex to search for the right docking site. Besides evaluating the geometric complementation, the free energy of each conformation also needs to be evaluated as part of the scoring function.

- In this dissertation, the energy computation is based on static biomolecular models. For dynamic models, it is expensive and unnecessary to re-compute all of the Born radii at each time step. Adaptivity needs to be considered as a modification of the current GB model.

- The CG models we have generated so far are at the same level. For proteins, there are certain regions, for example the active sites, where one needs detailed information, so efforts need to be put into develop multiscaled CG models.

Appendices

Appendix A

Smoothness of ASMS

Proof of Theorem 2.2.1: It is obvious that S is C^1 at the vertices. For the continuity at the midpoints of edges, let us consider the edge $\mathbf{v}_i\mathbf{v}_j$ in triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$. On the edge $\mathbf{v}_i\mathbf{v}_j$, $b_3 = 0$. So we may let $b_2 = t$ and $b_1 = 1 - t$. Then matrix T can be written as

$$T(t) = \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^{\mathrm{T}} \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^{\mathrm{T}} \\ (\mathbf{n}_i + t(\mathbf{n}_j - \mathbf{n}_i))^{\mathrm{T}} \end{pmatrix},$$

and

$$T^{-1} = \frac{1}{\det(T)} \left(B \times C,\ C \times A,\ A \times B\right),$$

where $A = \mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda)$, $B = \mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda)$ and $C(t) = \mathbf{n}_i + t(\mathbf{n}_j - \mathbf{n}_i)$. Therefore on the edge $\mathbf{v}_i\mathbf{v}_j$,

$$\begin{pmatrix} \frac{\partial F}{\partial b_1} \\ \frac{\partial F}{\partial b_2} \\ \frac{\partial F}{\partial \lambda} \end{pmatrix} = \begin{pmatrix} A^{\mathrm{T}} \\ B^{\mathrm{T}} \\ C^{\mathrm{T}} \end{pmatrix} (\mathbf{n}_i(1-t)^2 + \mathbf{n}_j t^2) + \begin{pmatrix} 3(b_{210}(\lambda) - b_{111}(\lambda)) \\ 3(b_{120}(\lambda) - b_{111}(\lambda)) \\ 1 \end{pmatrix} 2t(1-t).$$

The gradient of F on the edge $\mathbf{v}_i\mathbf{v}_j$ can be written as

$$\begin{aligned} \nabla F =& \mathbf{n}_i(1-t)^2 + \mathbf{n}_j t^2 + T^{-1} \begin{pmatrix} 3(b_{210}(\lambda) - b_{111}(\lambda)) \\ 3(b_{120}(\lambda) - b_{111}(\lambda)) \\ 1 \end{pmatrix} 2t(1-t) \\ =& \mathbf{n}_i(1-t)^2 + \mathbf{n}_j t^2 + \frac{2t(1-t)}{\det(T)(t)} \big[3(B \times C(t))(b_{210}(\lambda) - b_{111}(\lambda)) \\ &+ 3(C(t) \times A)(b_{120}(\lambda) - b_{111}(\lambda)) + A \times B\big]. \end{aligned} \tag{A.0.1}$$

When $t = \frac{1}{2}$, $C(\frac{1}{2}) = \mathbf{c}$, therefore

$$B \times C(t) + C(t) \times A = \mathbf{d}_3(\lambda).$$

Consider the function inside the square bracket of (A.0.1) into account, denoted as F_1:

$$F_1 = 3(B \times \mathbf{c})b_{210} + 3(\mathbf{c} \times A)(b_{120} + A \times B - 3(B \times \mathbf{c} + \mathbf{c} \times A)b_{111}. \quad \text{(A.0.2)}$$

Since on the edge $\mathbf{v}_i\mathbf{v}_j$, $b_{111}(\lambda) = b^{(3)}_{111}(\lambda)$, substituting (2.21) into (A.0.2), we get F_1 is 0. Therefore, at the midpoint

$$\nabla F = (\mathbf{n}_i + \mathbf{n}_j)/4. \quad \text{(A.0.3)}$$

So S is C^1 continuous at the midpoints of the edges. □

Proof of Theorem 2.2.2: It is obvious that S is C^1 within the triangles. By Theorem 2.2.1 we have already known that S is C^1 at the vertices and the midpoints of the edges. Here we only need to show S is C^1 at any points of the edges, let us consider the the edge $\mathbf{v}_i\mathbf{v}_j$ in the triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$.

Under the condition $\mathbf{n}_i \cdot (\mathbf{v}_i - \mathbf{v}_j) = \mathbf{n}_j \cdot (\mathbf{v}_j - \mathbf{v}_i)$, we have $b_{120}(\lambda) = b_{210}(\lambda)$, so (A.0.1) is written as

$$\nabla F = \mathbf{n}_i(1-t)^2 + \mathbf{n}_j t^2 + \frac{2t(1-t)}{\det(T(t))}[3(b_{210} - b_{111})(B-A) \times C + A \times B]. \quad \text{(A.0.4)}$$

Similar to (2.18), we define

$$\mathbf{d}_3(t, \lambda) = (B - A) \times C(t). \quad \text{(A.0.5)}$$

By (2.21) together with the facts that $b_{120}(\lambda) = b_{210}(\lambda)$ and $b_{111}(\lambda) = b^{(3)}_{111}(\lambda)$ on edge $\mathbf{v}_i\mathbf{v}_j$, we have

$$b_{210}(\lambda) - b_{111}(\lambda) = -\frac{\mathbf{d}_3^{\mathrm{T}}(\lambda)(A \times B)}{3||\mathbf{d}_3(\lambda)||^2}, \tag{A.0.6}$$

where $\mathbf{d}_3(\lambda)$ is defined in (2.20). Plug (A.0.5) and (A.0.6) in (A.0.4), we get

$$\begin{aligned}\nabla F = &\mathbf{n}_i(1-t)^2 + \mathbf{n}_j t^2 \\ &+ \frac{2t(1-t)}{||\mathbf{d}_3(\lambda)||^2}\left[\frac{||\mathbf{d}_3(\lambda)||^2 A \times B - \mathbf{d}_3(t,\lambda)\mathbf{d}_3^{\mathrm{T}}(\lambda)A \times B}{\det(T(t))}\right].\end{aligned} \tag{A.0.7}$$

Consider the function inside the square bracket of (A.0.7) and denote it as F_2. Our goal is to show that $F_2 = 0$. Since we have already known that when $t = \frac{1}{2}$, $F_2 = 0$, this prompts us to compute the derivative of F_2 with respect to t and see if the derivative is 0. We observe that both the numerator of the denominator of F_2 are linear in terms of t, so F_2 is of the form $\frac{at+b}{ct+d}$ with

$$\begin{aligned} a &= (\mathbf{n}_j - \mathbf{n}_i) \times (B - A)\mathbf{d}_3^{\mathrm{T}}(\lambda)A \times B, \\ b &= ||\mathbf{d}_3(\lambda)||^2 A \times B + \mathbf{n}_i \times (B - A)\mathbf{d}_3^{\mathrm{T}}(\lambda)A \times B, \\ c &= (\mathbf{n}_j - \mathbf{n}_i)^{\mathrm{T}}(A \times B), \\ d &= \mathbf{n}_i^{\mathrm{T}}(A \times B). \end{aligned}$$

In order to show $\frac{\partial F_2}{\partial t} = 0$, which is equivalent to show $N := ad - bc = 0$, we compute

$$\begin{aligned} N = &[\mathbf{n}_j \times (B - A)\mathbf{d}_3^{\mathrm{T}} A \times B]\mathbf{n}_i^{\mathrm{T}}(A \times B) \\ &-(||\mathbf{d}_3(\lambda)||^2 A \times B)(\mathbf{n}_j - \mathbf{n}_i) \times (B - A) \\ &-[\mathbf{n}_i \times (B - A)\mathbf{d}_3^{\mathrm{T}} A \times B]\mathbf{n}_j^{\mathrm{T}}(A \times B). \end{aligned} \tag{A.0.8}$$

Under the condition $\mathbf{n}_i \cdot (\mathbf{v}_i - \mathbf{v}_j) = \mathbf{n}_j \cdot (\mathbf{v}_j - \mathbf{v}_i)$, we have $(B-A)^{\mathrm{T}}\mathbf{c} = (\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda))^{\mathrm{T}}\mathbf{c} = 0$, where $\mathbf{c} = C(\frac{1}{2}, \frac{1}{2}, 0)$. Therefore

$$||\mathbf{d}_3(\lambda)||^2 = ((B-A)\times\mathbf{c})\cdot((B-A)\times\mathbf{c}) = ||\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda)||^2||\mathbf{c}||^2, \quad \text{(A.0.9)}$$

and

$$\begin{aligned}\mathbf{d}_3^{\mathrm{T}}(\lambda)A\times B &= \mathbf{d}_3^{\mathrm{T}}(\lambda)A\times(B-A)\\ &= ((B-A)\times\mathbf{c})\cdot(A\times(B-A)) = -\mathbf{c}^{\mathrm{T}}A||\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda)||^2.\end{aligned} \quad \text{(A.0.10)}$$

Plug (A.0.9) and (A.0.10) into (A.0.8) and divide both sides by $||\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda)||^2$, we get

$$\begin{aligned}F_3 :=& \frac{N}{||(\mathbf{v}_j - \mathbf{v}_i)(\lambda)||^2}\\ =& -\mathbf{n}_j\times(B-A)\mathbf{c}^{\mathrm{T}}A\mathbf{n}_i^{\mathrm{T}}(A\times B) - ||\mathbf{c}||^2 A\times B(\mathbf{n}_j - \mathbf{n}_i)^{\mathrm{T}}A\times B\\ &+ (\mathbf{n}_i\times(B-A)\mathbf{c}^{\mathrm{T}}A)\mathbf{n}_j^{\mathrm{T}}(A\times B)\\ =& [(\mathbf{c}^{\mathrm{T}}A\mathbf{n}_i - ||\mathbf{c}||^2 A)\times(B-A)]\mathbf{n}_j^{\mathrm{T}}(A\times B)\\ &+ [(||\mathbf{c}||^2 A - \mathbf{c}^{\mathrm{T}}A\mathbf{n}_j)\times(B-A)]\mathbf{n}_i^{\mathrm{T}}(A\times B).\end{aligned} \quad \text{(A.0.11)}$$

If $\mathbf{n}_i = \mathbf{n}_j$, (A.0.11) is 0. Now let us assume $\mathbf{n}_i \neq \mathbf{n}_j$. Recall that $\mathbf{c} = \frac{1}{2}(\mathbf{n}_i + \mathbf{n}_j)$. we define another vector $\mathbf{e} = \frac{1}{2}(\mathbf{n}_i - \mathbf{n}_j)$ and let $D = B - A$. Then $\mathbf{c}$ is orthogonal to $\mathbf{e}$ and D:

$$\mathbf{c}^{\mathrm{T}}\mathbf{e} = 0, \qquad \mathbf{c}^{\mathrm{T}}D = 0. \quad \text{(A.0.12)}$$

Furthermore

$$\mathbf{c}\times(D\times\mathbf{e}) = 0. \quad \text{(A.0.13)}$$

By the definition of $\mathbf{c}$ and $\mathbf{e}$,

$$\mathbf{n}_i = \mathbf{c} + \mathbf{e}, \qquad \mathbf{n}_j = \mathbf{c} - \mathbf{e}. \tag{A.0.14}$$

Substitute (A.0.14) into (A.0.11) and replace $A \times B$ with $A \times D$, we get

$$\begin{aligned} F_3 =& [\mathbf{c}^{\mathrm{T}} A(\mathbf{c}+\mathbf{e}) - ||\mathbf{c}||^2 A] \times D(\mathbf{c}-\mathbf{e})^{\mathrm{T}}(A \times D) \\ &+ [||\mathbf{c}||^2 A - \mathbf{c}^{\mathrm{T}} A(\mathbf{c}-\mathbf{e})] \times D(\mathbf{c}+\mathbf{e})^{\mathrm{T}}(A \times D) \\ =& 2\mathbf{c}^{\mathrm{T}} A(\mathbf{e} \times D)\mathbf{c}^{\mathrm{T}}(A \times D) \\ &- 2[\mathbf{c}^{\mathrm{T}} A\mathbf{c} - ||\mathbf{c}||^2 A] \times D\mathbf{e}^{\mathrm{T}}(A \times D). \end{aligned} \tag{A.0.15}$$

If $\mathbf{e}$ and D are linearly dependent, then $\mathbf{e} \times D = 0$, moreover $\mathbf{e}(A \times D) = 0$, which yields $F_3 = 0$. Otherwise, we introduce a new matrix

$$M = \begin{pmatrix} D^{\mathrm{T}} \\ \mathbf{c}^{\mathrm{T}} \\ \mathbf{e}^{\mathrm{T}} \end{pmatrix}.$$

Since $\mathbf{c}$, $\mathbf{e}$, and D are linearly independent, M is nonsingular. So F_3 (which is a vector) is equal to

$$\begin{aligned} & 2M^{-1} \begin{pmatrix} D^{\mathrm{T}} \\ \mathbf{c}^{\mathrm{T}} \\ \mathbf{e}^{\mathrm{T}} \end{pmatrix} \left(\mathbf{c}^{\mathrm{T}} A(\mathbf{e} \times D)\mathbf{c}^{\mathrm{T}}(A \times D) - [\mathbf{c}^{\mathrm{T}} A\mathbf{c} - ||\mathbf{c}||^2 A] \times D\mathbf{e}^{\mathrm{T}}(A \times D)\right) \\ =& -2M^{-1} \begin{pmatrix} 0 \\ (-\mathbf{c}^{\mathrm{T}} A\mathbf{c}^{\mathrm{T}}(\mathbf{e} \times D) - ||\mathbf{c}||^2 \mathbf{e}^{\mathrm{T}}(A \times D))\mathbf{c}^{\mathrm{T}}(A \times D) \\ (\mathbf{c}^{\mathrm{T}} A\mathbf{e}^{\mathrm{T}}(\mathbf{c} \times D) - ||\mathbf{c}||^2 \mathbf{e}^{\mathrm{T}}(A \times D))\mathbf{e}^{\mathrm{T}}(A \times D) \end{pmatrix} \\ =& -2M^{-1} \begin{pmatrix} 0 \\ (\mathbf{c}^{\mathrm{T}} A\mathbf{c}^{\mathrm{T}}(D \times \mathbf{e}) - ||\mathbf{c}||^2 A^{\mathrm{T}}(D \times \mathbf{e}))\mathbf{c}^{\mathrm{T}}(A \times D) \\ (\mathbf{c}^{\mathrm{T}} A\mathbf{c}^{\mathrm{T}}(D \times \mathbf{e}) - ||\mathbf{c}||^2 A^{\mathrm{T}}(D \times \mathbf{e}))\mathbf{e}^{\mathrm{T}}(A \times D) \end{pmatrix} \\ =& -2[\mathbf{c}^{\mathrm{T}} A\mathbf{c}^{\mathrm{T}}(D \times \mathbf{e}) - ||\mathbf{c}||^2 A^{\mathrm{T}}(D \times \mathbf{e})]M^{-1} \begin{pmatrix} 0 \\ \mathbf{c}^{\mathrm{T}}(A \times D) \\ \mathbf{e}^{\mathrm{T}}(A \times D) \end{pmatrix}. \end{aligned}$$

By the Lagrange's formula:

$$\mathbf{c}^{\mathrm{T}}A\mathbf{c}^{\mathrm{T}}(D\times\mathbf{e})-||\mathbf{c}||^2A^{\mathrm{T}}(D\times\mathbf{e})=(\mathbf{c}\times A)\cdot(\mathbf{c}\times(D\times\mathbf{e})), \tag{A.0.16}$$

and (A.0.13), (A.0.16) is zero and thus $F_3=0$. So far we have proved that F_2 is independent of t. Meanwhile in the proof of Theorem 2.2.1, we know that $F_2=0$ at $t=\frac{1}{2}$. Hence $F_2=0$ for all t and therefore on the edge $\mathbf{v}_i\mathbf{v}_j$, ∇F is

$$\nabla F=\mathbf{n}_i(1-t)^2+\mathbf{n}_jt^2.$$

So S is C^1 on the edges. □

Proof of Theorem 2.2.3: As same as the proof of Theorem 2.2.2, we only need to show that S is C^1 on the edge $\mathbf{v}_i\mathbf{v}_j$. In the proof of Theorem 2.2.1, we have already derived the gradient function on the edge $\mathbf{v}_i\mathbf{v}_j$ (A.0.1):

$$\begin{aligned}\nabla F =&\mathbf{n}_i(1-t)^2+\mathbf{n}_jt^2+\frac{2t(1-t)}{\det(T)(t)}[3(B\times C(t))(b_{210}(\lambda)-b_{111}(\lambda))\\&+3(C(t)\times A)(b_{120}(\lambda)-b_{111}(\lambda))+A\times B].\end{aligned}$$

Let

$$F_4=\frac{1}{\det(T)(t)}[3(B\times C(t))(b_{210}-b_{111})+3(C(t)\times A)(b_{120}-b_{111})+A\times B]. \tag{A.0.17}$$

Following the same idea of the proof the Theorem 2.2.2, we compute $\frac{\partial F_4}{\partial t}$. The numerator of $\frac{\partial F_4}{\partial t}$ is

$$\begin{aligned}&[3(B\times C'(t))(b_{210}-b_{111})+3(C'(t)\times A)(b_{120}-b_{111})\\&+A\times B]\det(T)-\det(T)'(t)[3(B\times C(t))(b_{210}-b_{111})\\&+3(C(t)\times A)(b_{120}-b_{111})+A\times B].\end{aligned} \tag{A.0.18}$$

Since

$$C'(t) = \mathbf{n}_j - \mathbf{n}_i,$$

and

$$\det(T)'(t) = (\mathbf{n}_j - \mathbf{n}_i)^{\mathrm{T}}(A \times B),$$

we get (A.0.18) is 0 when $\mathbf{n}_i = \mathbf{n}_j$. So F_4 is independent of t. By the proof of Theorem 2.2.1, $F_4 = 0$ at $t = \frac{1}{2}$. So $F_4 = 0$ for all t. So S is C^1 continuous. □

Appendix B

Fast Summation

B.1 Fast summation

The fast summation algorithm mainly deals with the summation of the form

$$G(\mathbf{x}_i) = \sum_{k=1}^{N} c_k g(\mathbf{x}_i - \mathbf{r}_k), \qquad i = 1, \ldots, M, \tag{B.1.1}$$

based on the nonequispaced fast Fourier transform (NFFT) [56]. In (3.4.3), the kernel function g is $\frac{1}{|\mathbf{x}|^4}$ which decays rapidly. Hence, we cut off the tail of g and assume that the support of g is bounded. Since the distance between $\mathbf{x}_i$ and $\mathbf{r}_k$ is no less than the smallest radius of the atoms, there is no singularity in g. Without loss of generality, we assume $\mathbf{x}-\mathbf{r}_k \in \Pi := [-\frac{1}{2}, \frac{1}{2}]^3 \backslash (-\epsilon, \epsilon)^3$. By duplicating g in the other blocks, g can be extended to be a periodic function of period one in $\mathbb{R}^3$ and this periodic function can be decomposed into the Fourier series:

$$g(\mathbf{x} - \mathbf{r}_k) = \sum_{\boldsymbol{\omega} \in I_\infty} g_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega} \cdot (\mathbf{x} - \mathbf{r}_k)}, \tag{B.1.2}$$

where $I_\infty := \{(\omega_1, \omega_2, \omega_3) \in \mathbb{Z}^3\}$ and $g_\omega = \int_\Pi g(\mathbf{x}) e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{x}} \, \mathrm{d}\mathbf{x}$. We approximate (B.1.2) by a truncated series:

$$g(\mathbf{x} - \mathbf{r}_k) \approx \sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} e^{2\pi i (\mathbf{x} - \mathbf{r}_k) \cdot \boldsymbol{\omega}}, \tag{B.1.3}$$

where $I_n := \{(\omega_1, \omega_2, \omega_3) \in \mathbb{Z}^3 : -\frac{n}{2} \leq \omega_i < \frac{n}{2}\}$. The Fourier coefficients g_ω are numerically computed as

$$g_{\boldsymbol{\omega}} = \frac{1}{n^3} \sum_{\mathbf{j} \in I_n} g(\frac{\mathbf{j}}{n}) e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{j}/n}, \qquad \boldsymbol{\omega} \in I_n. \tag{B.1.4}$$

By the fast Fourier transform (FFT) algorithm, eq (B.1.4) can be computed as fast as $O(n^3 \log n)$ time complexity.

Plug (B.1.3) into (B.1.1), we get

$$\begin{aligned} G(\mathbf{x}_i) &\approx \sum_{k=1}^{N} c_k \Big(\sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} e^{2\pi i (\mathbf{x}_i - \mathbf{x}_k) . \boldsymbol{\omega}} \Big) \\ &= \sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} \left(\sum_{k=1}^{N} c_k e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{r}_k} \right) e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}_i} \\ &= \sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} a_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}_i} \end{aligned} \tag{B.1.5}$$

$i = 1, \ldots, M$, where

$$a_{\boldsymbol{\omega}} = \sum_{k=1}^{N} c_k e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{r}_k}. \tag{B.1.6}$$

We compute (B.1.5) and (B.1.6) by the NFFT and NFFT^{T} with time complexity $O(n^3 \log n + m^3 M)$ and $O(n^3 \log n + m^3 N)$, respectively. Therefore the time complexity of computing (B.1.1) is $O(N + M + n^3 \log n)$, which significantly enhance the speed comparing with the trivial $O(MN)$ summation method if n which is the number of terms in the Fourier series is relatively small comparing with M and N. In the following sections, we will explain how we apply the fast summation algorithm to the electrostatic solvation energy and forces computations.

B.2 NFFT

The NFFT [57] is an algorithm for fast computation of multivariate discrete Fourier transforms for nonequispaced data in spacial domain (NDFT[1]). The NDFT[1] problem is to evaluate the trigonometric polynomials

$$G(\mathbf{x}_j) = \sum_{\boldsymbol{\omega} \in I_n} G_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}_j} \qquad j = 1, \ldots, M, \tag{B.2.1}$$

where $I_n = \{(\omega_1, \omega_2, \omega_3) \in \mathbb{Z}^3 : -\frac{n}{2} \leq \omega_i \leq \frac{n}{2}\}$. Without loss of generality, we assume $\mathbf{x}_j \in [-\frac{1}{2}, \frac{1}{2}]^3$. Instead of computing the summations in (B.2.1) directly, one can approximate G by a function $s(\mathbf{x})$ which is a linear combination of the shifted 1-periodic kernel function ξ:

$$s(\mathbf{x}) := \sum_{\mathbf{l} \in I_{\sigma n}} g_{\mathbf{l}} \xi(\mathbf{x} - \frac{\mathbf{l}}{\sigma n}), \tag{B.2.2}$$

where $I_{\sigma n} := \{(l_1, l_2, l_3) : l_i \in [-\frac{\sigma n}{2}, \frac{\sigma n}{2}] \cap \mathbb{Z}, \sigma > 1\}$ and $\frac{\mathbf{l}}{\sigma n} := \{(\frac{l_1}{\sigma n}, \frac{l_2}{\sigma n}, \frac{l_3}{\sigma n})\}$. The reason for $\sigma > 1$ is due to the error estimation discussed in section 3.4.3.3.

The kernel function ξ is defined as

$$\xi(\mathbf{x}) := \sum_{\mathbf{i} \in \mathbb{Z}^3} \xi_0(\mathbf{x} + \mathbf{i}), \qquad \text{where } \xi_0 \in L_2(\mathbb{R}^3).$$

Good candidates for ξ_0 include Gaussian, B-spline, sinc, and Kaiser-Bessel functions. Expand the periodic kernel function ξ by its Fourier series

$$\xi(\mathbf{x}) = \sum_{\boldsymbol{\omega} \in \mathbb{Z}^3} C_{\boldsymbol{\omega}}(\xi) e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}}, \tag{B.2.3}$$

with the Fourier coefficients

$$C_{\boldsymbol{\omega}}(\xi) := \int_{[-\frac{1}{2}, \frac{1}{2}]^3} \xi(\mathbf{x}) e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{x}} \, \mathrm{d}\mathbf{x} = \int_{\mathbb{R}^3} \xi_0(\mathbf{x}) e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{x}} \, \mathrm{d}\mathbf{x} = \hat{\xi}_0(\boldsymbol{\omega}).$$

Cut off the higher frequencies in (B.2.3), one can get

$$\xi(\mathbf{x}) = \left(\sum_{\boldsymbol{\omega} \in I_{\sigma n}} + \sum_{\boldsymbol{\omega} \in \mathbb{Z}^3 \setminus I_{\sigma n}} \right) C_{\boldsymbol{\omega}}(\xi) e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}} \approx \sum_{\boldsymbol{\omega} \in I_{\sigma n}} C_{\boldsymbol{\omega}}(\xi) e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}}. \tag{B.2.4}$$

Plug (B.2.4) into (B.2.2), we get

$$\begin{aligned} s(\mathbf{x}_j) &\approx \sum_{\mathbf{l} \in I_{\sigma n}} g_{\mathbf{l}} \sum_{\boldsymbol{\omega} \in I_{\sigma n}} C_{\boldsymbol{\omega}}(\xi) e^{2\pi i \boldsymbol{\omega} \cdot (\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n})} \\ &= \sum_{\boldsymbol{\omega} \in I_{\sigma n}} \tilde{G}_{\boldsymbol{\omega}} C_{\boldsymbol{\omega}}(\xi) e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}_j}, \end{aligned} \tag{B.2.5}$$

with the coefficients

$$\tilde{G}_{\boldsymbol{\omega}} := \sum_{\mathbf{l} \in I_{\sigma n}} g_{\mathbf{l}} e^{-2\pi i \boldsymbol{\omega} \cdot \frac{\mathbf{l}}{\sigma n}}. \tag{B.2.6}$$

By defining

$$\tilde{G}_{\boldsymbol{\omega}} := \begin{cases} \frac{G_{\boldsymbol{\omega}}}{C_{\boldsymbol{\omega}}(\xi)} & \text{for } \boldsymbol{\omega} \in I_n, \\ 0 & \text{for } \boldsymbol{\omega} \in I_{\sigma n} \setminus I_n, \end{cases} \tag{B.2.7}$$

one can immediately get

$$s(\mathbf{x}_j) \approx \sum_{\boldsymbol{\omega} \in I_n} G_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}_j} = G(\mathbf{x}_j). \tag{B.2.8}$$

The next problem is to compute $g_{\mathbf{l}}$. From (B.2.6), one can compute the coefficients $g_{\mathbf{l}}$ which are also coefficients in (B.2.8) by the discrete Fourier transform

$$g_{\mathbf{l}} = \frac{1}{\sigma^3 n^3} \sum_{\boldsymbol{\omega} \in I_{\sigma n}} \tilde{G}_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega} \cdot \frac{\mathbf{l}}{\sigma n}} = \frac{1}{\sigma^3 n^3} \sum_{\boldsymbol{\omega} \in I_n} \frac{G_{\boldsymbol{\omega}}}{C_{\boldsymbol{\omega}}(\xi)} e^{2\pi i \boldsymbol{\omega} \cdot \frac{\mathbf{l}}{\sigma n}}, \ \mathbf{l} \in I_{\sigma n}, \tag{B.2.9}$$

with complexity $O(n^3 \log n)$ by the FFT algorithm.

Since the function ξ drops very fast, one can further reduce the computation complexity of (B.2.8) by cutting off the tail of ξ. Define a function η_0:

$$\eta_0 := \xi_0(\mathbf{x})\chi_{[-\frac{m}{\sigma n},\frac{m}{\sigma n}]^3}(\mathbf{x}) \qquad \text{where } m \ll \sigma n, m \in \mathbb{N}.$$

Construct the one-periodic function η using the same way as constructing ξ:

$$\eta(\mathbf{x}) = \sum_{\mathbf{i}\in\mathbb{Z}^3} \eta_0(\mathbf{x}+\mathbf{i}).$$

Replacing ξ with η in (B.2.8), we obtain that

$$G(\mathbf{x}_j) \approx \sum_{\mathbf{l}\in I_{\sigma n,m}(\mathbf{x}_j)} g_{\mathbf{l}}\eta(\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n}), \tag{B.2.10}$$

where $I_{\sigma n,m}(\mathbf{x}_j) = \{(l_1,l_2,l_3) :\ \sigma n\mathbf{x}_{j,i} - m \le l_i \le \sigma n\mathbf{x}_{j,i} + m,\ i = 1,2,3\}$. There are at most $(2m+1)^3$ nonzero terms in (B.2.10). Therefore the complexity of evaluating (B.2.10) for $j = 1,\ldots,M$ is $O(m^3M)$. Adding the complexity of computing the coefficients $g_{\mathbf{l}}$, the overall complexity of NFFT algorithm is $O(n^3\log n + m^3M)$.

Remark B.2.1. If we reorganize the above equations, it is not hard to see that, in fact, (B.2.1) is approximately computed by the expression

$$G(\mathbf{x}_j) = \sum_{\boldsymbol{\omega}\in I_n} G_{\boldsymbol{\omega}} \left(\frac{1}{(\sigma n)^3 C_{\boldsymbol{\omega}}(\xi)} \sum_{\mathbf{l}\in I_{\sigma n}} \eta(\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n}) e^{2\pi i \boldsymbol{\omega}\cdot\frac{\mathbf{l}}{\sigma n}} \right). \tag{B.2.11}$$

From a linear algebra point of view, equation (B.2.11) can be written as the product of a matrix and a vector. For example, for a one dimensional NFFT, (B.2.11) is equivalent to

$$\mathbf{g} = \Xi\, F\, D\, \hat{\mathbf{g}} \tag{B.2.12}$$

with vectors

$$\mathbf{g} := [G(x_i)]_{i=1}^{M}, \quad \hat{\mathbf{g}} := [G_\omega]_{\omega=-\frac{n}{2}}^{\frac{n}{2}-1}.$$

Ξ is a sparse matrix

$$\Xi := \left[\eta(x_i - \frac{l_j}{\sigma n})\right]_{M\times\sigma n},$$

F is the classical Fourier matrix

$$F := \left[e^{2\pi i\omega_j \frac{l_i}{\sigma n}}\right]_{\sigma n\times n},$$

and D is an $n \times n$ diagonal matrix with the iith element being $\frac{1}{\sigma n C_{\omega_i}(\xi)}$. For a multi-dimensional NFFT, it is the same as the 1D case as long as one orders the indices of the multi-dimension into one dimension.

As discussed in [57], in the first approximation (B.2.8), we see that ξ is equal to f after its high frequencies in the Fourier series are cut off. Hence the error introduced in (B.2.8) which is known as the *aliasing error* is

$$\begin{aligned} E\text{NFFT}^1 :&= \sum_{\mathbf{l}\in I_{\sigma n}} g_{\mathbf{l}}\xi(\mathbf{x}_j - \frac{\mathbf{l}}{\sigma n}) - f(\mathbf{x}_j) \\ &= \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} \sum_{\boldsymbol{\omega}\in I_{\sigma n}} \tilde{G}_{\boldsymbol{\omega}+\mathbf{i}\sigma n} C_{\boldsymbol{\omega}+\mathbf{i}\sigma n}(\xi) e^{2\pi i(\boldsymbol{\omega}+\mathbf{i}\sigma n)\cdot\mathbf{x}_j}. \end{aligned} \tag{B.2.13}$$

Note that from (B.2.6), we have the condition $\tilde{G}_{\boldsymbol{\omega}+\mathbf{i}\sigma n} = \tilde{G}_{\boldsymbol{\omega}}$, for $\mathbf{i} \in \mathbb{Z}^3$ and $\boldsymbol{\omega} \in I_{\sigma n}$. By the definition (B.2.7), one obtains

$$|E\text{NFFT}^1| \le \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} \sum_{\boldsymbol{\omega}\in I_n} |G_{\boldsymbol{\omega}} \frac{C_{\boldsymbol{\omega}+\mathbf{i}\sigma n}(\xi)}{C_{\boldsymbol{\omega}}(\xi)}|. \tag{B.2.14}$$

Let $||\hat{G}||_1 = \sum_{\boldsymbol{\omega}\in I_n} |G_{\boldsymbol{\omega}}|$. Then

$$|E\text{NFFT}^1| \le ||\hat{G}||_1 \max_{\boldsymbol{\omega}\in I_n} \sum_{\mathbf{i}\in\mathbb{Z}^3\setminus\{0\}} |\frac{C_{\boldsymbol{\omega}+\mathbf{i}\sigma n}(\xi)}{C_{\boldsymbol{\omega}}(\xi)}|. \tag{B.2.15}$$

In the second approximation (B.2.10), since ξ is replaced by η, the so caused error, known as the *truncation error*, is

$$\begin{aligned} E\text{NFFT}^2 :&= \sum_{\mathbf{l}\in I_{\sigma n}} g_{\mathbf{l}}\xi(\mathbf{x}-\frac{\mathbf{l}}{\sigma n}) - \sum_{\mathbf{l}\in I_{\sigma n,m}} g_{\mathbf{l}}\eta(\mathbf{x}-\frac{\mathbf{l}}{\sigma n}) \\ &= \sum_{\mathbf{l}\in I_{\sigma n}\setminus I_{\sigma n,m}} g_{\mathbf{l}}[\xi(\mathbf{x}-\frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}-\frac{\mathbf{l}}{\sigma n})] \\ &= \sum_{\mathbf{l}\in I_{\sigma n}\setminus I_{\sigma n,m}} \frac{1}{\sigma^3 n^3} \sum_{\boldsymbol{\omega}\in I_n} \frac{G_{\boldsymbol{\omega}}}{C_{\boldsymbol{\omega}}(\xi)} e^{2\pi i\boldsymbol{\omega}\cdot\frac{\mathbf{l}}{\sigma n}} [\xi(\mathbf{x}-\frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}-\frac{\mathbf{l}}{\sigma n})]. \end{aligned} \tag{B.2.16}$$

Thus

$$\begin{aligned} |E\text{NFFT}^2| &\le \frac{1}{\sigma^3 n^3} \sum_{\mathbf{l}\in I_{\sigma n}} |\frac{G_{\boldsymbol{\omega}}}{C_{\boldsymbol{\omega}}(\xi)}[\xi(\mathbf{x}-\frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}-\frac{\mathbf{l}}{\sigma n})]| \\ &\le \frac{1}{\sigma^3 n^3} \max_{\boldsymbol{\omega}\in I_n}(C_{\boldsymbol{\omega}}^{-1}(\xi))||\hat{G}||_1 \sum_{\mathbf{l}\in I_{\sigma n}} |\xi(\mathbf{x}-\frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{x}-\frac{\mathbf{l}}{\sigma n})|. \end{aligned} \tag{B.2.17}$$

B.3 NFFT$^{\mathrm{T}}$

The NFFT$^{\mathrm{T}}$ algorithm deals with the fast computation of multivariate discrete Fourier transforms for nonequispaced data in frequency domain (NDFT^2):

$$a(\boldsymbol{\omega}) = \sum_{k=1}^{N} c_k e^{-2\pi i\boldsymbol{\omega}\cdot\mathbf{r}_k}, \qquad \boldsymbol{\omega}\in I_n. \tag{B.3.1}$$

Define a function

$$A(\mathbf{x}) := \sum_{k=1}^{N} c_k \xi(\mathbf{x}-\mathbf{r}_k), \tag{B.3.2}$$

where ξ is defined as same as in Appendix B.2. The Fourier series of $A(\mathbf{x})$ is:

$$A(\mathbf{x}) = \sum_{\boldsymbol{\omega}\in\mathbb{Z}^3} C_{\boldsymbol{\omega}}(A) e^{2\pi i \boldsymbol{\omega}\cdot\mathbf{x}}. \tag{B.3.3}$$

On the other hand,

$$\sum_{k=1}^{N} c_k \xi(\mathbf{x}-\mathbf{r}_k) = \sum_{k=1}^{N} c_k \sum_{\boldsymbol{\omega}\in\mathbb{Z}^3} C_{\boldsymbol{\omega}}(\xi) e^{2\pi i \boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{r}_k)}. \tag{B.3.4}$$

Hence we get the relationship of Fourier coefficients of A and ξ:

$$C_{\boldsymbol{\omega}}(A) = \sum_{k=1}^{N} c_k e^{-2\pi i \boldsymbol{\omega}\cdot\mathbf{r}_k} C_{\boldsymbol{\omega}}(\xi), \qquad \boldsymbol{\omega}\in\mathbb{Z}^3. \tag{B.3.5}$$

Comparing (B.3.5) with (B.3.1) one obtains

$$a(\boldsymbol{\omega}) = \frac{C_{\boldsymbol{\omega}}(A)}{C_{\boldsymbol{\omega}}(\xi)}, \qquad \boldsymbol{\omega}\in I_n. \tag{B.3.6}$$

It remains to compute $C_{\boldsymbol{\omega}}(A)$. By definition,

$$\begin{aligned} C_{\boldsymbol{\omega}}(A) &= \int_{[-\frac{1}{2},\frac{1}{2}]^3} A(\mathbf{x}) e^{-2\pi i \boldsymbol{\omega}\cdot\mathbf{x}}\,\mathrm{d}\mathbf{x} \\ &= \int_{[-\frac{1}{2},\frac{1}{2}]^3} \left(\sum_{k=1}^{N} c_k \xi(\mathbf{x}-\mathbf{r}_k)\right) e^{-2\pi i \boldsymbol{\omega}\cdot\mathbf{x}}\,\mathrm{d}\mathbf{x} \\ &= \sum_{k=1}^{N} c_k \int_{[-\frac{1}{2},\frac{1}{2}]^3} \xi(\mathbf{x}-\mathbf{r}_k) e^{-2\pi i \boldsymbol{\omega}\cdot\mathbf{x}}\,\mathrm{d}\mathbf{x}. \end{aligned} \tag{B.3.7}$$

Discretizing the integration in (B.3.7) by the left rectangular rule leads to

$$a(\boldsymbol{\omega}) \approx \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \sum_{k=1}^{N} c_k \frac{1}{(\sigma n)^3} \sum_{\mathbf{l}\in I_{\sigma n}} \xi\left(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k\right) e^{-2\pi i \boldsymbol{\omega}\cdot\frac{\mathbf{l}}{\sigma n}}. \tag{B.3.8}$$

Replacing ξ with η yields

$$a(\boldsymbol{\omega}) \approx \frac{1}{(\sigma n)^3} \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \sum_{\mathbf{l}\in I_{\sigma n}} \hat{g}_{\mathbf{l}} e^{-2\pi i \boldsymbol{\omega}\cdot\frac{\mathbf{l}}{\sigma n}}, \tag{B.3.9}$$

where

$$\hat{g}_{\mathbf{l}} := \sum_{k=1}^{N} c_k \eta(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k), \qquad \mathbf{l} \in I_{\sigma n}. \tag{B.3.10}$$

To compute $\hat{g}_{\mathbf{l}}$, if one scans the $\mathbf{r}_k$ list, then for each $\mathbf{r}_k$ there are at most $(2m+1)^3$ grid points ($\mathbf{l}$) that contribute nonzero η. Hence, the complexity of computing $\hat{g}_{\mathbf{l}}$ is $O(m^3 N)$. After computing $\hat{g}_{\mathbf{l}}$ one can easily evaluate (B.3.9) by the FFT algorithm at the complexity of $O(n^3 \log n)$. Lastly the complexity of computing (B.3.6) is $O(n^3)$. So the overall complexity of the NFFT$^{\mathrm{T}}$ algorithm is $O(m^3 N + n^3 \log n)$.

Remark B.3.1. Similar to the NFFT algorithm, we may write the one-line formula for computing (B.3.1) by the NFFT$^{\mathrm{T}}$:

$$a(\boldsymbol{\omega}) = \sum_{k=1}^{N} c_k \left(\frac{1}{(\sigma n)^3 C_{\boldsymbol{\omega}}(\xi)} \sum_{\mathbf{l} \in I_{\sigma n}} \eta(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k) e^{-2\pi i \boldsymbol{\omega} \cdot \frac{\mathbf{l}}{\sigma n}} \right), \tag{B.3.11}$$

which in one dimension is equivalent to the linear system:

$$\hat{\mathbf{a}} = D^{\mathrm{T}} F^* \, \Xi^{\mathrm{T}} \, \mathbf{c} \tag{B.3.12}$$

with vectors

$$\hat{\mathbf{a}} := [a(\omega)]_{\omega=-\frac{n}{2}}^{\frac{n}{2}}, \quad \mathbf{c} := [c_k]_{k=1}^{N}.$$

Matrix Ξ is similar to that defined in Appendix B.2

$$\Xi := \left[\eta(\frac{l_j}{\sigma n} - r_i) \right]_{N \times \sigma n}.$$

F^* is the conjugate transpose of the Fourier matrix F, and D is the same as that defined in Appendix B.2. From the matrix expression, we see why the algorithm is called the "transpose" of NFFT.

Let $E_{\boldsymbol{\omega}}$ designate the error of $a(\boldsymbol{\omega})$. $E_{\boldsymbol{\omega}}$ can also split into the *aliasing error* $E^1_{\boldsymbol{\omega}}$ introduced in (B.3.8) and the *truncation error* $E^2_{\boldsymbol{\omega}}$ introduced in (B.3.9):$E_{\boldsymbol{\omega}} = E^1_{\boldsymbol{\omega}} + E^2_{\boldsymbol{\omega}}$, for $\boldsymbol{\omega} \in I_{\sigma n}$. In (B.3.8), expand ξ to its Fourier series, we get

$$\begin{aligned} E^1_{\boldsymbol{\omega}} &= a(\boldsymbol{\omega}) - \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \sum_{k=1}^{N} c_k \frac{1}{(\sigma n)^3} \sum_{\mathbf{l} \in I_{\sigma n}} \left(\sum_{\mathbf{j} \in \mathbb{Z}^3} C_{\mathbf{j}}(\xi) e^{2\pi i (\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k) \cdot \mathbf{j}} \right) e^{-2\pi i \boldsymbol{\omega} \cdot \frac{\mathbf{l}}{\sigma n}} \\ &= a(\boldsymbol{\omega}) - \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \sum_{k=1}^{N} c_k \sum_{\mathbf{j} \in \mathbb{Z}^3} C_{\mathbf{j}}(\xi) e^{-2\pi i \mathbf{r}_k \cdot \mathbf{j}} \frac{1}{(\sigma n)^3} \sum_{\mathbf{l} \in I_{\sigma n}} e^{2\pi i (\mathbf{j} - \boldsymbol{\omega}) \cdot \frac{\mathbf{l}}{\sigma n}}. \end{aligned}$$

Since

$$\frac{1}{(\sigma n)^3} \sum_{\mathbf{l} \in I_{\sigma n}} e^{2\pi i (\mathbf{j} - \boldsymbol{\omega}) \cdot \frac{\mathbf{l}}{\sigma n}} = \begin{cases} 1, & \text{if } \mathbf{j} - \boldsymbol{\omega} = \mathbf{i}\sigma n,\ \mathbf{i} \in \mathbb{Z}^3, \\ 0, & \text{otherwise,} \end{cases}$$

$$E^1_{\boldsymbol{\omega}} = a(\boldsymbol{\omega}) - \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \sum_{k=1}^{N} c_k \sum_{\mathbf{i} \in \mathbb{Z}^3} C_{\boldsymbol{\omega} + \mathbf{i}\sigma n}(\xi) e^{-2\pi i \mathbf{r}_k \cdot (\boldsymbol{\omega} + \mathbf{i}\sigma n)}. \tag{B.3.13}$$

By (B.3.1),

$$E^1_{\boldsymbol{\omega}} = \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \sum_{k=1}^{N} c_k \sum_{\mathbf{i} \in \mathbb{Z}^3 \setminus \{0\}} C_{\boldsymbol{\omega} + \mathbf{i}\sigma n}(\xi) e^{-2\pi i \mathbf{r}_k \cdot (\boldsymbol{\omega} + \mathbf{i}\sigma n)}. \tag{B.3.14}$$

Define $||\mathbf{c}||_1 = \sum_{k=1}^{N} |c_k|$. Then we have

$$|E^1_{\boldsymbol{\omega}}| \leq ||\mathbf{c}||_1 \sum_{\mathbf{i} \in \mathbb{Z}^3 \setminus \{0\}} \frac{C_{\boldsymbol{\omega} + \mathbf{i}\sigma n}(\xi)}{C_{\boldsymbol{\omega}}(\xi)}. \tag{B.3.15}$$

In (B.3.9), the truncation error

$$E^2_{\boldsymbol{\omega}} = \sum_{k=1}^{N} c_k \left(\frac{1}{(\sigma n)^3 C_{\boldsymbol{\omega}}(\xi)} \sum_{\mathbf{l} \in I_{\sigma n}} [\xi(\mathbf{r}_k - \frac{\mathbf{l}}{\sigma n}) - \eta(\mathbf{r}_k - \frac{\mathbf{l}}{\sigma n})] e^{-2\pi i \boldsymbol{\omega} \cdot \frac{\mathbf{l}}{\sigma n}} \right), \tag{B.3.16}$$

which has the bound

$$|E^2_{\boldsymbol{\omega}}| \leq \frac{1}{(\sigma n)^3}||\mathbf{c}||_1 \frac{1}{C_{\boldsymbol{\omega}}(\xi)} \max_k \sum_{\mathbf{l}\in I_{\sigma n}} |\xi(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k) - \eta(\frac{\mathbf{l}}{\sigma n} - \mathbf{r}_k)|. \qquad \text{(B.3.17)}$$

Appendix C

Continuity of the integrand function in the Born radii calculation

As defined in Section 3.4.3.1,

$$f = \frac{(\mathbf{r} - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4}, \tag{C.0.1}$$

where $\mathbf{r} \neq \mathbf{x}_i$ and $\mathbf{n} = \frac{\nabla F}{||\nabla F||}$ with F defined in (2.4). $\mathbf{r}(b_1, b_2, \lambda)$ is simply defined in (2.22). In this appendix, we mainly discuss the continuity of $\mathbf{n}$. As derived in Chapter 2,

$$\nabla F = \mathfrak{T}^{-1} \left(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \frac{\partial F}{\partial \lambda} \right)^{\mathrm{T}} \tag{C.0.2}$$

where $\mathfrak{T}$ is a nonsingular matrix. Hence $\mathbf{n}$ is well defined. Consider $(\frac{\partial \mathbf{n}}{\partial b_1}, \frac{\partial \mathbf{n}}{\partial b_2})$:

$$\left(\frac{\partial \mathbf{n}}{\partial b_1}, \frac{\partial \mathbf{n}}{\partial b_2} \right) = \begin{pmatrix} F_{xx} & F_{xy} & F_{xz} \\ F_{xy} & F_{yy} & F_{yz} \\ F_{xz} & F_{yz} & F_{zz} \end{pmatrix} \begin{pmatrix} \frac{\partial x}{\partial b_1} & \frac{\partial x}{\partial b_2} \\ \frac{\partial y}{\partial b_1} & \frac{\partial y}{\partial b_2} \\ \frac{\partial z}{\partial b_1} & \frac{\partial z}{\partial b_2} \end{pmatrix}$$

. Let $\nu = \left(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \frac{\partial F}{\partial \lambda} \right)^{\mathrm{T}}$. We have

$$\begin{pmatrix} F_{xx} & F_{xy} & F_{xz} \\ F_{xy} & F_{yy} & F_{yz} \\ F_{xz} & F_{yz} & F_{zz} \end{pmatrix} = \begin{pmatrix} \mathfrak{T}_x & \mathfrak{T}_y & \mathfrak{T}_z \end{pmatrix} \begin{pmatrix} \nu & & \\ & \nu & \\ & & \nu \end{pmatrix} + \mathfrak{T} M \mathfrak{T}^{\mathrm{T}} \tag{C.0.3}$$

where $F_{xy} = \frac{\partial^2 F}{\partial x \partial y}$, $\mathfrak{T}_x = \frac{\partial \mathfrak{T}}{\partial x}$, and

$$M = \begin{pmatrix} F_{b_1 b_1} & F_{b_1 b_2} & F_{b_1 \lambda} \\ F_{b_1 b_2} & F_{b_2 b_2} & F_{b_2 \lambda} \\ F_{b_1 \lambda} & F_{b_2 \lambda} & F_{\lambda\lambda} \end{pmatrix}. \tag{C.0.4}$$

To show $\mathcal{T}$ is differentiable, we take the first row of $\mathcal{T}$ and compute its derivative with respect to x, i.e. $(\frac{\partial^2 b_1}{\partial x^2}\ \frac{\partial^2 b_2}{\partial x^2}\ \frac{\partial^2 \lambda}{\partial x^2})$ as an example. We write (2.22) in the form of

$$\begin{cases} x = x(b_1, b_2, \lambda), \\ y = y(b_1, b_2, \lambda), \\ z = z(b_1, b_2, \lambda). \end{cases} \tag{C.0.5}$$

Take the second derivatives of both sides of (C.0.5) with respect to x, we get

$$0 = C_f + \frac{\partial x}{\partial b_1}\frac{\partial^2 b_1}{\partial x^2} + \frac{\partial x}{\partial b_2}\frac{\partial^2 b_2}{\partial x^2} + \frac{\partial x}{\partial \lambda}\frac{\partial^2 \lambda}{\partial x^2}, \tag{C.0.6}$$

$$0 = C_g + \frac{\partial y}{\partial b_1}\frac{\partial^2 b_1}{\partial x^2} + \frac{\partial y}{\partial b_2}\frac{\partial^2 b_2}{\partial x^2} + \frac{\partial y}{\partial \lambda}\frac{\partial^2 \lambda}{\partial x^2}, \tag{C.0.7}$$

$$0 = C_h + \frac{\partial z}{\partial b_1}\frac{\partial^2 b_1}{\partial x^2} + \frac{\partial z}{\partial b_2}\frac{\partial^2 b_2}{\partial x^2} + \frac{\partial z}{\partial \lambda}\frac{\partial^2 \lambda}{\partial x^2}. \tag{C.0.8}$$

where

$$C_f = \begin{pmatrix} \frac{\partial b_1}{\partial x} \\ \frac{\partial b_2}{\partial x} \\ \frac{\partial \lambda}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial^2 x}{\partial b_1^2}\frac{\partial b_1}{\partial x} + \frac{\partial^2 x}{\partial b_1 \partial b_2}\frac{\partial b_2}{\partial x} + \frac{\partial^2 x}{\partial b_1 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 x}{\partial b_1 \partial b_2}\frac{\partial b_1}{\partial x} + \frac{\partial^2 x}{\partial b_2^2}\frac{\partial b_2}{\partial x} + \frac{\partial^2 x}{\partial b_2 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 x}{\partial b_1 \partial \lambda}\frac{\partial b_1}{\partial x} + \frac{\partial^2 x}{\partial b_2 \partial \lambda}\frac{\partial b_2}{\partial x} + \frac{\partial^2 x}{\partial \lambda^2}\frac{\partial \lambda}{\partial x} \end{pmatrix},$$

$$C_g = \begin{pmatrix} \frac{\partial b_1}{\partial x} \\ \frac{\partial b_2}{\partial x} \\ \frac{\partial \lambda}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial^2 y}{\partial b_1^2}\frac{\partial b_1}{\partial x} + \frac{\partial^2 y}{\partial b_1 \partial b_2}\frac{\partial b_2}{\partial x} + \frac{\partial^2 y}{\partial b_1 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 y}{\partial b_1 \partial b_2}\frac{\partial b_1}{\partial x} + \frac{\partial^2 y}{\partial b_2^2}\frac{\partial b_2}{\partial x} + \frac{\partial^2 y}{\partial b_2 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 y}{\partial b_1 \partial \lambda}\frac{\partial b_1}{\partial x} + \frac{\partial^2 y}{\partial b_2 \partial \lambda}\frac{\partial b_2}{\partial x} + \frac{\partial^2 y}{\partial \lambda^2}\frac{\partial \lambda}{\partial x} \end{pmatrix},$$

$$C_h = \begin{pmatrix} \frac{\partial b_1}{\partial x} \\ \frac{\partial b_2}{\partial x} \\ \frac{\partial \lambda}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial^2 z}{\partial b_1^2}\frac{\partial b_1}{\partial x} + \frac{\partial^2 z}{\partial b_1 \partial b_2}\frac{\partial b_2}{\partial x} + \frac{\partial^2 z}{\partial b_1 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 z}{\partial b_1 \partial b_2}\frac{\partial b_1}{\partial x} + \frac{\partial^2 z}{\partial b_2^2}\frac{\partial b_2}{\partial x} + \frac{\partial^2 z}{\partial b_2 \partial \lambda}\frac{\partial \lambda}{\partial x} \\ \frac{\partial^2 z}{\partial b_1 \partial \lambda}\frac{\partial b_1}{\partial x} + \frac{\partial^2 z}{\partial b_2 \partial \lambda}\frac{\partial b_2}{\partial x} + \frac{\partial^2 z}{\partial \lambda^2}\frac{\partial \lambda}{\partial x} \end{pmatrix}.$$

So we get

$$\begin{pmatrix} \frac{\partial^2 b_1}{\partial x^2} \\ \frac{\partial^2 b_2}{\partial x^2} \\ \frac{\partial^2 \lambda}{\partial x^2} \end{pmatrix} = \mathcal{T} \begin{pmatrix} -C_f \\ -C_g \\ -C_h \end{pmatrix}. \tag{C.0.9}$$

Using the same method, we can get the other rows of $\frac{\partial \mathcal{T}}{\partial x}$, matrices $\mathcal{T}_y$ and $\mathcal{T}_z$ by changing C_f, C_g, C_h in (C.0.9). Therefore $\mathcal{T}$ is differentiable. Similarly,

we can compute the higher order derivatives of $\mathcal{T}$ and prove that $\mathcal{T} \in C^\infty$, thus prove $F \in C^\infty(\Omega_0)$, where Ω_0 defined in Section 3.4.3.1 is the canonical triangle. Therefore, as defined in (C.0.1), $f \in C^\infty(\Omega_0)$.

Bibliography

[1] D. Dunavant. High degree efficient symmetrical Gaussian quadrature rules for the triangle. *Int. J. Numer. Meth. Engng.*, 21:1129–1148, 1985.

[2] M. Feig, A. Onufriev, M. Lee, W. Im, D. Case, and C. Brooks, III. Performance comparison of generalized Born and Poisson methods in the calculation of electrostatic solvation energies for protein structures. *J. Comput. Chem.*, 25:265–284, 2004.

[3] M. Levitt, M. Hirshberg, R. Sharon, and V. Daggett. Potential energy function and parameters for simulations of the molecular dynamics of proteins and nucleic acids in solution. *Comp. Phys. Comm.*, 91:215–231, 1995.

[4] W. Cornell, P. Cieplak, C. Bayly, I. Gould, K. Merz, Jr., D. Ferguson, D. Spellmeyer, T. Fox, J. Caldwell, and P. Kollman. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Am. Chem. Soc.*, 117:5179–5197, 1995.

[5] J. Gao, K. Kuczera, B. Tidor, and M. Karplus. Hidden thermodynamics of mutant proteins: a molecular dynamics analysis. *Science*, 244:1069–1072, 1989.

[6] B. Roux and T. Simonson. Implicit solvent models. *Biophys. Chem.*, 78:1–20, 1999.

[7] R. Pierotti. A scaled particle theory of aqueous and nonaqueous solutions. *Chem. Rev.*, 76:717–726, 1976.

[8] O. Sinanoglu. Surface tension down to molecular dimensions and microthermodynamic surface areas of molecules or clusters. *J. Chem. Phys.*, 75:463–468, 1981.

[9] J. Tomasi and M. Persico. Molecular interactions in solution: An overview of methods based on continuous distributions of the solvent. *Chem. Rev.*, 94:2027–2094, 1994.

[10] J. Madura, J. Briggs, R. Wade, M. Davis, B. Luty, A. Ilin, J. Antosiewicz, M. Gilson, B. Bagheri, L. Scott, and J.A. McCammon. Electrostatics and diffusion of molecules in solution: simulations with the University of Houston Brownian Dynamics program. *Comput. Phys. Comm.*, 91:57–95, 1995.

[11] B. Honig and A. Nicholls. Classical electrostatics in biology and chemistry. *Science*, 268:1144–1149, 1995.

[12] C. Cortis and R. Friesner. An automatic three-dimensional finite element mesh generation system for the Poisson-Boltzmann equation. *J. Comput. Chem.*, 18:1570–1590, 1997.

[13] N. Baker, M. Holst, and F. Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems. *J. Comput. Chem.*, 21:1343–1352, 2000.

[14] M. Totrov and R. Abagyan. Rapid boundary element solvation electrostatics calculations in folding simulations: Successful folding of a 23-residue peptide. *Biopolyms Peptide Sci*, 60:124–133, 2001.

[15] B. Lu, X. Cheng, and J.A. McCammon. New-version-fast-multipole-method accelerated electrostatic calculations in biomolecular systems. *J. Chem. Phys.*, 226:1348–1366, 2007.

[16] D. Tobias. Electrostatics calculations: Recent methodological advances and applications to membranes. *Curr. Opin. Struct. Biol.*, 11:253–C261, 2001.

[17] B. Lee and F. Richards. The interpretation of protein structure: estimation of static accessiblilty. *J. Mol. Biol.*, 55:379–400, 1971.

[18] M. Connolly. Analytical molecular surface calculation. *J. Appl. Cryst.*, 16:548–558, 1983.

[19] F. Richards. Areas, volumes, packing, and protein structure. *Annu. Rev. Biophys. Bioeng.*, 6:151–176, 1977.

[20] M. Lee, M. Feig, F. Salsbury, and C. Brooks, III. New analytic approximation to the standard molecular volume definition and its application to generalized Born calculations. *J. Comput. Chem.*, 24:1348–1356, 2003.

[21] J. Grant and B. Pickup. A Gaussian description of molecular shape. *J. Phys. Chem.*, 99:3503–3510, 1995.

[22] M. Lee, F. Salsbury, and C. Brooks, III. Novel generalized Born methods. *J. Chem. Phys.*, 116:10606–10614, 2002.

[23] C. Bajaj, J. Castrillon-Candas, V. Siddavanahalli, and Z. Xu. Compressed representations of macromolecular structures and properties. *Structure*, 13:463–471, 2005.

[24] C. Bajaj and V. Siddavanahalli. Fast error-bounded surfaces and derivatives computation for volumetric particle data. ICES Technical Report TR-06-06, 2006.

[25] C. Bajaj, H. Lee, R. Merkert, and V. Pascucci. NURBS based B-rep models from macromolecules and their properties. *In Proceedings of Fourth Symposium on Solid Modeling and Applications*, pages 217–228, 1997.

[26] H. Edelsbrunner. Deformable smooth surface design. *Discrete Computational Geometry*, 21:87–115, 1999.

[27] H. Cheng and X. Shi. Guaranteed quality triangulation of molecular skin surfaces. *IEEE Visualization*, pages 481–488, 2004.

[28] H. Cheng and X. Shi. Quality mesh generation for molecular skin surfaces using restricted union of balls. *IEEE Visualization*, pages 51–58, 2005.

[29] W. Dahmen. Smooth piecewise quadratic surfaces. In T. Lyche and L. Schumaker, editors, *Mathematical methods in computer aided geometric design*, pages 181–193. Academic Press, Boston, 1989.

[30] B. Guo. *Modeling arbitrary smooth objects with algebraic surfaces*. PhD thesis, Cornell University, 1991.

[31] W. Dahmen and T-M. Thamm-Schaar. Cubicoids: modeling and visualization. *Computer Aided Geometric Design*, 10:89–108, 1993.

[32] C. Bajaj, J. Chen, and G. Xu. Modeling with cubic A-patches. *ACM Transactions on Graphics*, 14:103–133, 1995.

[33] N. Akkiraju and H. Edelsbrunner. Triangulating the surface of a molecule. *Discrete Applied Mathematics*, 71:5–22, 1996.

[34] P. Laug and H. Borouchaki. Molecular surface modeling and meshing. *Engineering with Computers*, 18:199–210, 2002.

[35] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Computer Aided Geometric Design*, 23:510–530, 2006.

[36] C. Bajaj and V. Siddavanahalli. An adaptive grid based method for computing molecular surfaces and properties. ICES Technical Report TR-06-57, 2006.

[37] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *Proceedings of ACM SIGGRAPH*, pages 339–346, 2002.

[38] M. Garland and P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. *IEEE Visualization*, pages 263–270, 1998.

[39] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. Texmol: Interactive visual exploration of large flexible multi-component molecular complexes. *Proc. of the Annual IEEE Visualization Conference*, pages 243–250, 2004.

[40] G. Nielson. The side-vertex method for interpolation in triangles. *J. Apprrox. Theory*, 25:318–336, 1979.

[41] D. Eisenberg and A. Mclachlan. Solvation energy in protein folding and binding. *Nature (London)*, 319:199–203, 1986.

[42] M. Gilson, M. Davis, B. Luty, and J.A. McCammon. Computation of electrostatic forces on solvated molecules using the Poisson-Boltzmann equation. *J. Phys. Chem.*, 97:3591–3600, 1993.

[43] R. Hermann. Theory of hydrophobic bonding. II. Correlation of hydrocarbon solubility in water with solvent cavity surface area. *J. Phys. Chem.*, 76:2754–2759, 1972.

[44] K. Sharp. Incorporating solvent and ion screening into molecular dynamics using the finite-difference Poisson-Boltzmann method. *J. Comput. Chem.*, 12:454–468, 1991.

[45] T. Simonson and A. Bruenger. Solvation free energies estimated from macroscopic continuum theory: An accuracy assessment. *J. Phys. Chem.*, 98:4683 – 4694, 1994.

[46] W. Still, A. Tempczyk, R. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J. Am. Chem. Soc.*, 112:6127–6129, 1990.

[47] K. Lum, D. Chandler, and J. Weeks. Hydrophobicity at small and large length scales. *J. Phys. Chem. B*, 103:4570–4577, 1999.

[48] A. Shih, I. Denisov, J. Phillips, S. Sligar, and K. Schulten. Molecular dynamics simulations of discoidal bilayers assembled from truncated human lipoproteins. *Biophys. J.*, 88:548–556, 2005.

[49] M. Born. Volume and hydration warmth of ions. *Zeitschrift Fur Physik*, 1:45–48, 1920.

[50] J. Srinivasan, M. Trevathan, P. Beroza, and D. Case. Application of a pairwise generalized Born model to proteins and nucleic acids: inclusion of salt effects. *Theor. Chem. Accts.*, 101:426–434, 1999.

[51] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Chem. Phys.*, 73:325–348, 1987.

[52] D. Bashford and D. Case. Generalized Born models of macromolecular solvation effects. *Annu. Rev. Phys. Chem.*, 51:129–152, 2000.

[53] A. Ghosh, C. Rapp, and R. Friesner. Generalized Born model based on a surface integral formulation. *J. Phys. Chem. B*, 102:10983–10990, 1998.

[54] W. Im, M. Lee, and C. Brooks, III. Generalized Born model with a simple smoothing function. *J. Comput. Chem.*, 24:1691–1702, 2003.

[55] M. Scarsi, J. Apostolakis, and A. Caflisch. Continuum electrostatic energies of macromolecules in aqueous solutions. *J. Phys. Chem. A*, 101:8098–8106, 1997.

[56] D. Potts and G. Steidl. Fast summation at nonequispaced knots by NFFTs. *SIAM J. Sci. Comput.*, 24:2013–2037, 2003.

[57] D. Potts, G. Steidl, and M. Tasche. *Fast Fourier transforms for nonequispaced data: A tutorial, in Modern Samplling Theory: mathematics and Applications*, pages 247–270. Birkhauser, 2001.

[58] D. Case, T. Cheatham, III, T. Darden, H. Gohlke, R. Luo, K. Merz, Jr., A. Onufriev, C. Simmerling, B. Wang, and R. Woods. The Amber biomolecular simulation programs. *J. Comput. Chem.*, 26:1668–1688, 2005.

[59] W. Im, D. Beglov, and B. Roux. Continuum solvation model: Computation of electrostatic forces from numerical solutions to the Poisson-Boltzmann equation. *Comput. Phys. Comm.*, 111:59–75, 1998.

[60] Z. Yu, M. Jacobson, and R. Friesner. What role do surfaces play in GB models? A new-generation of surface-generalized Born model based on a

novel Gaussian surface fro biomolecules. *J. Comput. Chem.*, 27:72–89, 2006.

[61] J. Kubelka, J. Hofrichter, and W. Eaton. The protein folding 'speed limit'. *Curr. Opin. Struct. Biol.*, 14:76–88, 2004.

[62] F. Pizzitutti, M. Marchi, and D. Borgis. Coarse-graining the accessible surface and the electrostatics of proteins for protein-protein interactions. *J. Chem. Theory and Comput.*, 3:1867–1876, 2007.

[63] V. Tozzini. Coarse-grained models for proteins. *Curr. Opin. Struct. Biol.*, 15:144–150, 2005.

[64] J. Trylska, V. Tozzini, and J.A. McCammon. Exploring global motions and correlations in the ribosome. *Biophys. J.*, 89:1455–1463, 2005.

[65] V. Tozzini and J.A. McCammon. A coarse grained model for the dynamics of flap opening in HIV-1 protease. *Chem. Phys. Lett.*, 413:123–128, 2005.

[66] A. Arkhipov, P. Freddolino, and K. Schulten. Stability and dynamics of virus capsids described by coarse-grained modeling. *Structure*, 14:1767–1777, 2006.

[67] M. Stevens. Coarse-grained simulations of lipid bilayers. *J. Chem. Phys.*, 121:11942–11948, 2004.

[68] A. Shih, A. Arkhipov, P. Freddolino, and K. Schulten. Coarse grained protein-lipid model with application to lipoprotein particles. *J. Phys. Chem. B*, 110:3674–3684, 2006.

[69] P. Bond and M. Sansom. Insertion and assembly of membrane proteins via simulation. 128:2697–2704, 2006.

[70] M. Cieplak and T. Hoang. Coarse grained description of protein folding. *Phys. Rev. E*, 58:3589–3596, 1998.

[71] R. G. Larson, L. E. Scriven, and H. T. Davis. Monte Carlo simulation of model amphiphile-oilwater systems. *J. Chem. Phys.*, 83:2411–2420, 1985.

[72] J. Onuchic, Z. Schulten, and P. Wolynes. Theory of protein folding: The energy landscape perspective. *Annu. Rev. Phys. Chem.*, 48:545–600, 1997.

[73] S. Marrink, A. de Vries, and A. Mark. Coarse grained model for semi-quantitative lipid simulations. *J. Phys. Chem. B*, 108:750–760, 2004.

[74] B. Smit, P. Hilbers, K. Esselink, L. Rupert, N. van Os, and A. Schlijper. Computer simulations of a water/oil interface in the presence of micelles. *Nature*, 348:624–625, 1990.

[75] J. Shelley, M. Shelley, R. Reeder, S. Bandyopadhyay, and M. Klein. A coarse grain model for phospholipid simulations. *J. Phys. Chem. B*, 105:4464–4470, 2005.

[76] J. Shelley, M. Shelley, R. Reeder, S. Bandyopadhyay, P. Moore, and M. Klein. Simulations of phospholipids using a coarse grain model. *J. Phys. Chem. B*, 105:9785–9792, 2001.

[77] Y. Ueeda, H. Taketomi, and N. Gō. Studies on protein folding, unfolding and fluctuations by computer simulation. a three-dimensional lattice model of lysozyme. *Biopolymers*, 17:1531–1548, 1978.

[78] M. Cheung, A. Garcia, and J. Onuchic. Protein folding mediated by solvation: water expulsion and formation of the hydrophobic core occur after structural collapse. *Proc. Natl. Acad. Sci. USA*, 99:685–690, 2000.

[79] C. Clementi, A. Garcia, and J. Onuchic. Interplay among tertiary contacts, secondary structure formation and side chain packing in the protein folding mechanism: all-atom representation study of protein L. *J. Mol. Biol.*, 326:933–954, 2003.

[80] V. Tozzini, W. Rocchia, and J.A. McCammon. Mapping all-atom models onto one-bead coarse grained models: general properties and applications to a minimal polypeptide model. *J. Chem. Theory Comput.*, 2:667–673, 2006.

[81] V. Tozzini, J. Trylska, C. Chang, and J.A. McCammon. Flap opening dynamics in HIV-1 protease explored with a coarse-grained model. *J. Struc. Biol.*, 157:606–615, 2007.

[82] I. Bahar and R. Jernigan. Inter-residue potentials in globular proteins and the dominance of highly specific hydrophilic interactions at close separation. *J. Mol. Biol.*, 266:195–214, 1997.

[83] N.-V. Buchete, J. Straub, and D. Thirumalai. Anisotropic coarse-grained statistical potentials improve the ability to identify nativelike protein structures. *J. Chem. Phys.*, 118:7658–7671, 2003.

[84] N. Kurt N, T. Haliloglu T, and C. Schiffer. Structure -based prediction of potential binding and non binding peptides to HIV-1 protease. *Biophys J.*, 85:853–863, 2003.

[85] N.-V. Buchete, J. Straub, and D. Thirumalai. Orientational potentials extracted from protein structures imporve native fold recognition. *Protein Sci.*, 13:862–874, 2004.

[86] N. Basdevant, D. Borgis, and T. Ha-Duong. A coarse-grained protein-protein potential derived from an all-atom force field. *J. Phys. Chem. B*, 111:9390–9399, 2007.

[87] M. Zacharias. Protein-protein docking with a reduced protein model accounting for side-chain flexibility. *Protein Sci.*, 12:1271–1282, 2003.

[88] A. Smith and C. Hall. α-Helix formation: Discontinuous molecular dynamics on an intermediate-resolution protein model. *Proteins: Structure, function, and genetics*, 44:344–360, 2001.

[89] A. Arkhipov, P. Freddolino, K. Imada, K. Namba, and K. Schulten. Coarse-grained molecular dynamics simulations of a rotating bacterial flagellum. *Biophy. J.*, 91:4589–4597, 2006.

[90] P. A. Golubkov and P. Ren. Generalized coarse-grained model based on point multipole and Gay-Berne potentials. *J. Chem. Phys.*, 125:064103, 2006.

[91] S. Izvekov and G. Voth. A multiscale coarse-graining method for biomolecular systems. *J. Phys. Chem. B*, 109:2469–2473, 2005.

[92] S. Izvekov and G. Voth. Multiscale coarse graining of liquid-state systems. *J. Chem. Phys.*, 123:134105, 2005.

[93] D. Reith, M. Ptz, and F. Muller-Plathe. Deriving effective mesoscale potentials from atomistic simulations. *J. Comput. Chem.*, 24:1624–1636, 2003.

[94] T. Murtola, E. Falck, M. Patra, M. Karttunen, and I. Vattulainena. Coarse-grained model for phospholipid/cholesterol bilayer. *J. Chem. Phys.*, 121:9156–9165, 2004.

[95] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, 2007.

[96] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11:431–441, 1963.

Vita

Wenqi Zhao was born in Tianjin, China on April 29th 1980, the daughter of Daqian Zhao and Jinxiu Gong. She was admitted to Nankai University, Tianjin, in 1998. After she received the Bachelor of Science degree in Computational Mathematics in 2002, she entered the Graduate School of the University of Texas at Austin and enrolled in the Computational and Applied Mathematics program. In 2004, she received her Master of Science in Computational and Applied Mathematics.

Permanent address: 205 Zhongshan Road
Tianjin, China 300141

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.